توضیح سوال اول :

برای تشخیص اعداد زوج از یک حلقه استفاده می‌کنیم که طول عدد وارده را طی کند و ارقام را یکی یکی چک کند و اعداد زوج را برگزیند. به دلیل نوع داده‌های ورودی، عدد به صورت استرینگ (رشته‌ای) دریافت می‌شود.

در این کد، از بولین پیشوندی برای کنترل چاپ ستاره‌ها استفاده می‌کنیم. به این صورت که پس از چاپ عدد اول، ستاره‌ای چاپ نمی‌شود و بعد از آن، قبل از چاپ هر عدد زوج برگزیده شده، ستاره چاپ می‌شود. این کار برای جلوگیری از چاپ اضافی ستاره‌ها انجام می‌شود.

FastAPIرا ایمپورت می‌کنیم. سپس سه متد مختلف برای دریافت ورودی تعریف کردیمQuery Parameters :، Path Parameters، و.Request Body

متد Query Parameters ورودی را به صورت پارامترهای URL دریافت می‌کند. این متد با استفاده از دستور GETورودی عددی را به صورت رشته دریافت می‌کند و اعداد زوج را با ستاره‌ها جدا می‌کند.

متد Path Parameters ورودی را به عنوان قسمتی از مسیر URL دریافت می‌کند. این متد نیز از دستور GET استفاده می‌کند و ورودی عددی را به عنوان قسمتی از URL دریافت می‌کند و اعداد زوج را با ستاره‌ها جدا می‌کند.

در نهایت، برای دریافت ورودی به عنوان بدنه درخواست، از متد Request Body استفاده کردیم. این متد از دستور POST استفاده می‌کند و ورودی عددی را به عنوان بدنه درخواست دریافت می‌کند و اعداد زوج را با ستاره‌ها جدا می‌کند. برای این کار، مدل داده‌ای مناسبی تعریف کردیم که ورودی را اعتبارسنجی می‌کند.

برای اجرای سرورFastAPI ، از دستور uvicorn استفاده کردیم. این دستور سرور محلی را اجرا کرده و امکان ارسال درخواست‌ها بهAPI ها را فراهم می‌کند.

برای بررسی و تست متدها، ازURL های مربوطه استفاده کردیم Query Parameters .و Path Parameters از URLهای مشخصی استفاده کرده و Request Body نیز با استفاده از ابزارهایی مانند Postman یا Insomnia قابل تست است. این کد اعداد زوج را از یک رشته عددی دریافت کرده و بین آنها ستاره قرار می‌دهد

.

توضیح سوال دوم :

تابع فاکتوریل را تعریف میکنیم و مقدار ان را بازگردانی میکنیم

حاصل را در یک متغیر به اسم ریزالت ذخیره میکنیم ، الگوی صورت و مخرج کسر را پیدا کرده و با استفاده از حلقه الگو رو پیاده میکنیم

سپس صورت را بر مخرج تقسیم میکنیم و حاصل که مقدار اولیه ان صفر است با هربار تکرار حلقه با مقدار حاصل حلقه برای هرتکرار یا همان حاصل جمله اِن الگو جمع میشود

در اخر علامتی که در ابتدا مثبت تعریف کردیم با قرار دادن مقدار یک در متغیر ساین را قرینه میکنیم تا به صورت تناوبی جمع و تفریق عبارت ادامه یابد.

لازم به ذکر است برای صفر نشدن مخرج مطابق با شرایط الگو شرط تعیین میکنیم که اگر مخرج صفر شد کل عبارت صفر بشود تا در محاسبات نقشی نداشته باشد و علامت های تناوبی مثبت منفی درمیان را بهم نزد (اگر از دستور کانتنیو استفاده کنیم علامت ساین عوض میشد وحاصل متفاوت است !)

چون ورودی نداشتیم علی‌رغم اینکه می‌شد باز هم با سه متد یک نتیجه واحد را نشان دهیم، برای راحتی کار فقط از یک متد (path_parameter) استفاده کردیم که URL برای اجرای آن به صورت زیر است:

http://127.0.0.1:8000/result

توضیح سوال سوم :

ابتدا کتابخانه FastAPI را ایمپورت کردیم. سپس یک برنامه FastAPI ایجاد کردیم.

تابعی به نام find_numbers تعریف کردیم که اعداد مورد نظر را پیدا می‌کند. این تابع با استفاده از یک حلقه در بازه 1000 تا 9999 اعداد را به صورت استرینگ پیمایش کرده، جمع دو رقم اول و حاصل ضرب دو رقم آخر را محاسبه می‌کند. اگر این دو مقدار برابر باشند، آن عدد به لیست نتایج اضافه می‌شود. در نهایت، این تابع لیست نتایج را بازمی‌گرداند.

سپس یک API تعریف کردیم که نتایج را برمی‌گرداند. چون ورودی نداشتیم علی‌رغم اینکه می‌شد باز هم با سه متد یک نتیجه واحد را نشان دهیم، برای راحتی کار فقط از یک متد استفاده کردیم URL .برای اجرای این متد به صورت زیر است:

http://127.0.0.1:8000/result

توضیح سوال چهارم :

تابعی تعریف میکنیم که پارامتر ان اعداد هستند ابتدا کتابخانه FastAPI را ایمپورت کردیم. سپس یک برنامه FastAPIایجاد کردیم.

تابعی تعریف کردیم که پارامتر آن اعداد هستند. اعداد را به صورت رشته‌ای پیمایش می‌کنیم و اگر یکی از ارقام زوج نباشد، مقدار Falseبازمی‌گردانیم. در غیر این صورت، مقدار Trueرا بازمی‌گردانیم.

در تابع اصلی، یک حلقه برای بازه مورد نظر (100 تا 999) ایجاد کردیم و شرطی برقرار کردیم که اگر اعداد ما در تابعی که قبل تعریف کردیم مقدار Trueبرگرداند، آن‌ها را چاپ کنیم. چون از لیست استفاده نمی‌کنیم، در چاپ از endو فاصله استفاده کردیم تا اعداد مورد نظر مرتب چاپ شوند.

چون ورودی نداشتیم علی‌رغم اینکه می‌شد باز هم با سه متد یک نتیجه واحد را نشان دهیم، برای راحتی کار فقط از یک متد استفاده کردیم URL .برای اجرای این متد به صورت زیر است:

http://127.0.0.1:8000/result

توضیح سوال پنجم :

این کد یک API برای تولید الگوی اعداد مثلثی را ایجاد می‌کند. ابتدا کتابخانه‌های مورد نیاز FastAPI و Pydanticرا وارد می‌کنیم ; Pydantic .برای تعریف مدل داده‌ای مورد استفاده قرار می‌گیرد

. مدل داده‌ای NumberInputرا برای دریافت ورودی به عنوان بدنه درخواست تعریف می‌کنیم. این مدل شامل یک فیلد عددی nاست.

تابعی به نام generate_patternتعریف می‌کنیم که پارامتر آن nاست. این تابع با استفاده از حلقه‌های تودرتو، اعداد را به صورت الگوی اضافه شدن به خود تا توان دوم همان عدد چاپ می‌کند و نتیجه را به صورت یک لیست از لیست‌ها بازمی‌گرداند

و در اخر سه متد مختلف برای دریافت ورودی تعریف می‌کنیم

توضیح سوال ششم :

برای محاسبه امار از کتابخانه typing استفاده میکنیم
تابعی به نام statics_needed تعریف می‌کنیم که پارامتر آن یک لیست از اعداد اعشاری است. این تابع حداکثر، حداقل، میانگین، و انحراف استاندارد اعداد را محاسبه و بازگردانی می‌کند.

مدل داده‌ای NumberInputرا برای دریافت ورودی به عنوان بدنه درخواست تعریف می‌کنیم. این مدل شامل یک فیلد لیست از اعداد اعشاری به نام numbersاست.

و در اخر سه متد مختلف برای دریافت ورودی تعریف می‌کنیم

توضیح سوال هفتم :

همه موارد مانند سوال قبلی با تفاوت اینکه اینبار چهار تابع مختلف برای محاسبه حداکثر، حداقل، میانگین، و انحراف استاندارد اعداد تعریف می‌کنیم.

توضیح سوال هشتم :

کتابخانه ها مانند دو سوال قبلی مورد استفاده قرار میگیرند و درادامه

دو تابع مختلف برای پردازش اعداد تعریف می‌کنیم.

این تابع f1، بزرگترین رقم موجود در رشته عدد را پیدا می‌کند و برمی‌گرداند و تابع f2 بزرگترین رقم پیدا شده را از رشته عدد حذف می‌کند و نتیجه را برمی‌گرداند.

مدل داده‌ای NumberInput را برای دریافت ورودی به عنوان بدنه درخواست تعریف می‌کنیم. این مدل شامل یک فیلد رشته‌ای به نام number است.

در آخر سه متد مختلف برای دریافت ورودی تعریف می‌کنیم

نکات مهم برای توضیحات هر سوال که لازم به کدنویسی داشت :

مورد اول : سوال هایی که ورودی نداشتند به جای 3 متد تنها با متد **pathparameters** نوشته شده اند که اگر میخواستیم میتوانستیم از دومتد دیگر نیز استفاده کنیم به عنوان مثال اگر کد تنها یک خروجی واحد داشته باشد مثل سوال چهارم میتوانیم با دستور زیر متد **query** را نیز در ان استفاده کنیم :

```
@app.get("/Home")
async def home():
    return {"message": "Welcome to the Home Page"}
@app.get("/result")
async def calculate_path():
    numbers = find_numbers()
```

```
return {"numbers": numbers}
@app.get("/Home/result")
async def calculate_query(result: str = Query(...)):
    if result == "Result":
        numbers = find_numbers()
        return {"numbers": numbers}
    else:
        return {"error": "Invalid query parameter"}
```

که **url** برای کواری پارامتر به این شکل خواهد بود :

http://127.0.0.1:8000/Home/result?result=Result

مورد دوم :

استفاده از **async** به دلیل این است که سرور بتواند چندین درخواست رو همزمان پردازش کند. این باعث می شود که وقتی چند نفر همزمان درخواست ارسال میکنند ، سرور لازم نباشد منتظر بماند تا یک درخواست انجام بشود و بعدی شروع بشود . به این ترتیب، سرور سریع‌تر و بهینه‌تر کار می کند و می تواند چندین کار را همزمان انجام بدهد

مورد سوم :

دستور اجرا در ترمینال به صورت زیر انجام میشود :

uvicorn “نام فایل " : "نام متغیری که برنامه فست ای پی آی را نگه میدارد **–reload**

که طبق قرارداد برنامه نویسی در کدهایی که اراعه شده متغیر با نام **app** برنامه ی **fastapi** را نگه میدارد با سینتکس **app = fastapi()**

Uvicorn file:app – reload

توضیح سوال نهم :

به طور کلی متغیر هایی که درون ساختار یک تابع فرعی بوجود می ایند لوکال وریبلس و متغیر هایی که درون بدنه اصلی کد و تابع مِین بوجود می ایند گلوبال وریبلز میگویند بنابراین

در مثال پی1 :

در تابع اف ، لوکال وریبل با نام اس دارای یک استرینگ در خود است و با صدا زدن تابع اف استرینگ موجود در متغیر درون تابع فرعی چاپ خواهد شد زیرا دستور چاپ در خود تابع قرار دارد

درمثال پی2:

در تابع اف لوکال وریبل با نام اس تعریف شده و دارای استرینگی درون خود است و دستور چاپ در بدنه خود تابع نیز وجود دارد جلوتر در تابع مین تابع صدا زده شده و دستور پرینت متغیر اس نیز نوشته شده

اما خروجی تنها چاپ قسمتی است که تابع دستور پرینت ان را درخود دارد

و متغیر اس در تابع مین تعریف نشده بنابراین دستور چاپ ان در این تابع معنایی ندارد زیرا اس یک لوکال وریبل است نه گلوبال وریبل و تنها در بدنه تابع فرعی اف تعریف شده است

در مثال پی3:

در اینجا با استفاده از کلمه کلیدی گلوبال تابع اف گلوبال وریبل را تغییر میدهد و زمانی که اف فراخوانی میشود ، متغیر اس را به استرینگی که در بدنه خود تابع فرعی به متغیر اس داده شده تغییر میدهد و ان را چاپ میکند

بدین ترتیب چاپ متغیر اس حتی با تعریف در بدنه تابع اصلی و مین نیز دستخوش تغییر میشود و دوبار همان استرینگ درون تابع فرعی چاپ میشود زیرا مقدار متغیر گلوبال به وسیله تابع فرعی تغییر کرده است.

مثال پی4:

در این مثال دوباره متغیری با نام اس درون تابع قرار دارد که دستور چاپ ان نیز در بدنه خود تابع فرعی نیز وجود دارد اما تفاوت با مثال قبلی این است که متغیر اس در بدنه تابع اصلی یا همان تابع مِین هم تعریف شده است و اینبار هم با صدا زدن تابع و هم با دستور پرینت مستقیم در تابع مین برای اقدام به چاپ متغیر اس

ما دو خروجی خواهیم داشت

جالب توجه است که متغیری با نام اس در تابع فرعی تعریف شده محتوای متفاوتی نسبت به متغیری با همان نام که در تابع مِین تعریف شده دارد یعنی اس هم یک گلوبال وریبل است و هم در تابع فرعی بعنوان یک لوکال وریبل شناخته میشود و با این وجود در تابع اصلی مقدار ان بازخوانی و چاپ میشود و با صدا زدن تابع مقدار استرینگ موجود در اس درون تابع چاپ میشود

توضیح سوال دهم :

این تابع از دو قسمت شرط پایه که شامل ان است که اگر عدد مورد نظر برابر با 1 بود مقدار 1 برگردانده شود و شرطی بازگشتی شامل این که اگر هر عددی غیر از یک وارد شد یکی از ان کم بشود سپس فاکتوریلش گرفته بشود و در خود عدد ضرب بشود و سپس مقدار جدید بازخوانی بشود

دراجرای کد با ورودی 4 :

چون 4 با یک برابر نیست :

مقداری بازگشتی میشود 4 * 3! = 4 * 6 = 24

بدین ترتیب فاکتوریل 4 برابر شد با 4 ضرب در فاکتوریل سه که مساوی با 24 است

هیراد طولابی — 1403414