خانه فناوری تهران

جزوه تخصصی | بخش توسعه وب و بک‌اند

از پایتون تا توسعه وب با فریم‌ورک جنگو

معماری کلاینت–سرور، تفاوت فرانت‌اند و بک‌اند، طراحی UI/UX و مسیر ورود حرفه‌ای به توسعه نرم‌افزار

Python
Django
Client–Server
Frontend & Backend
Career Roadmap

ارائه بخش تخصصی: خانم فرشته داوودیان

با همراهی خانم حنانه رحمت‌زهی

برگرفته از وبینار «مسیر آموزش بعد از پایتون»

FANAVARI.CO

خلاصه جامع جلسه

مروری بر محتوای آموزشی

این جزوه بر پایه بخش آموزشی وبینار «مسیر آموزش بعد از پایتون» تدوین شده است. خانم فرشته داوودیان با همراهی خانم حنانه رحمت‌زهی، مسیر توسعه نرم‌افزار را با تمرکز بر تفاوت برنامه‌های خط فرمان، رابط‌های گرافیکی و وب‌اپلیکیشن‌ها تشریح کرده‌اند. بخش محوری آموزش به معماری کلاینت-سرور، مفهوم فرانت‌اند و بک‌اند و نقش فریم‌ورک Django در توسعه سمت سرور با پایتون اختصاص دارد. همچنین تفاوت طراحی UI و UX، امکان توسعه وب‌اپ‌های هوشمند و توصیه‌های حرفه‌ای درباره تمرکز تخصصی، ساخت نمونه‌کار، فعالیت در GitHub و LinkedIn و همکاری تیمی مطرح شده‌اند. در نسخه تدوین‌شده، مثال‌ها و نکات تکمیلی برای افزایش دقت فنی و استفاده مستقل دانشجو از مطالب افزوده شده است.

موضوعات و محورهای کلیدی

1. **از خط فرمان تا محصول کاربردی**
   تفاوت اجرای برنامه‌های متنی، نرم‌افزارهای گرافیکی و وب‌اپلیکیشن‌ها و نحوه انتخاب مناسب‌ترین بستر.

2. **معماری کلاینت-سرور و توسعه بک‌اند**
   جریان درخواست و پاسخ، فرانت‌اند، بک‌اند و نقش Django در توسعه سامانه‌های مبتنی بر پایتون.

3. **تفکیک UI، UX و توسعه فرانت‌اند**
   شناخت تفاوت طراحی تجربه، ظاهر رابط و پیاده‌سازی واقعی آن در مرورگر.

4. **وب‌اپلیکیشن و هوش مصنوعی**
   نحوه اتصال پروژه Django به سرویس‌ها و مدل‌های AI و الزامات تبدیل نمونه‌کار به محصول.

5. **مسیر شغلی، گیت‌هاب و لینکدین**
   تمرکز در یک شاخه، مستندسازی، همکاری تیمی و ارائه نمونه‌کار برای فرصت‌های حرفه‌ای.

مناسب برای: دانشجویان پایتون، علاقه‌مندان توسعه بک‌اند با Django و افرادی که می‌خواهند تفاوت طراحی رابط کاربری، توسعه وب و ساخت محصول مبتنی بر هوش مصنوعی را درک کنند.

راهنمای مطالعه جزوه

این جزوه بر اساس گفتار آموزشی جلسه تنظیم شده است. گفت‌وگوهای مرتبط با قطع و وصل صدا، هماهنگی ارائه و بخش‌های غیرآموزشی حذف شده‌اند. مثال‌ها، تمرین‌ها و توضیحات تکمیلی برای رفع ابهام‌های فنی و تقویت یادگیری اضافه شده‌اند؛ این افزوده‌ها نقل‌قول مستقیم مدرس محسوب نمی‌شوند.

پیش‌نیاز پیشنهادی: شناخت مقدماتی متغیرها، شرط‌ها، حلقه‌ها، تابع‌ها و فایل‌ها در پایتون. آشنایی اولیه با HTML و HTTP مفید است، اما در فصل‌ها مفاهیم ضروری توضیح داده شده‌اند.

فهرست مطالب

01 از پایتون به کدام نوع نرم‌افزار برسیم؟ 4

02 ساخت رابط گرافیکی دسکتاپ با پایتون 6

03 معماری کلاینت–سرور و مسیر یک درخواست وب 8

04 فرانت‌اند، بک‌اند، UI و UX؛ مرزها و همکاری‌ها 10

05 فریم‌ورک چیست و Django چه نقشی دارد؟ 12

06 ساخت یک وب‌اپلیکیشن واقعی با Django 14

07 وب‌اپلیکیشن، PWA و اتصال به پروژه‌های هوش مصنوعی 16

08 نقشه راه تخصصی و ورود حرفه‌ای به بازار کار 18

09 هوش مصنوعی، وایب‌کدینگ و آمادگی محیط عملیاتی 20

10 تمرین‌ها، چک‌لیست و واژه‌نامه 22

شماره صفحات در این نسخه به‌صورت خودکار در خروجی PDF محاسبه شده‌اند.

فصل 01

از پایتون به کدام نوع نرم‌افزار برسیم؟

۱. پایتون، شروع مسیر توسعه نرم‌افزار

یادگیری پایتون به شما امکان می‌دهد مسئله‌ای را به مجموعه‌ای از دستورها تبدیل کنید. خروجی این برنامه‌ها می‌تواند از یک اسکریپت ساده تا یک نرم‌افزار دارای رابط گرافیکی یا یک سرویس تحت وب متفاوت باشد. در جلسه، خانم داوودیان از همین نقطه شروع کردند: دانشجو پس از آموختن دستورات پایتون چگونه محصولی می‌سازد که افراد دیگر بتوانند از آن استفاده کنند؟

۲. سه شکل رایج اجرای برنامه

نوع برنامه	محل تعامل کاربر	نمونه کاربرد	ابزار یا فناوری
خط فرمان (CLI)	ترمینال یا پنجره Command Prompt	تغییر نام گروهی فایل‌ها، گزارش‌گیری	Python و ابزارهای استاندارد آن
نرم‌افزار دسکتاپ (GUI)	پنجره، دکمه، فرم و منو	ابزار حسابداری یا مدیریت فایل	Tkinter، PyQt / PySide
وب‌اپلیکیشن	مرورگر وب، گاهی اپ موبایل	سامانه آموزشی، فروشگاه، CRM	HTML، CSS، JavaScript و Django

نکته مهم این است که این دسته‌ها از نظر رابط کاربری و محل اجرا متفاوت‌اند؛ خود زبان پایتون ذاتاً فقط «زبان بک‌اند» نیست. از پایتون در اسکریپت‌ها، داده، دسکتاپ و سمت سرور نیز استفاده می‌شود.

۳. برنامه خط فرمان چیست؟

برنامه خط فرمان معمولاً ورودی را از آرگومان‌ها یا متن دریافت می‌کند و خروجی را به‌صورت نوشته ارائه می‌دهد. نبود دکمه و پنجره به معنی ناقص‌بودن برنامه نیست. بسیاری از ابزارهای حرفه‌ای توسعه، مدیریت سرور و اتوماسیون از خط فرمان استفاده می‌کنند.

```
name = input("Your name: ")
print(f"Hello, {name}!")
```

نمونه کد آموزشی است و زبان متن رابط می‌تواند تغییر کند.

۴. انتخاب روش مناسب برای مسئله

- اگر ابزار مخصوص یک تیم فنی است و سرعت اجرای دستورات مهم‌تر از ظاهر آن است، CLI می‌تواند کافی باشد.
- اگر کاربر روی رایانه به یک ابزار محلی با فرم‌ها و پنجره‌ها نیاز دارد، GUI دسکتاپ انتخاب مناسبی است.
- اگر کاربران باید از دستگاه‌ها و مکان‌های مختلف به یک سامانه مشترک دسترسی داشته باشند، توسعه وب می‌تواند گزینه مناسب‌تری باشد.

> **پیام آموزشی:** از ابتدا مشخص کنید مخاطب برنامه چه کسی است، از کجا باید از آن استفاده کند و چه نوع ارتباطی با سامانه خواهد داشت. ابزار را بعد از تعریف مسئله انتخاب کنید.

فصل 02

ساخت رابط گرافیکی دسکتاپ با پایتون

۱. مفهوم GUI

رابط کاربری گرافیکی یا Graphical User Interface مجموعه عناصر قابل مشاهده و قابل تعامل مانند پنجره، دکمه، فیلد متن، جدول و منوست. در چنین برنامه‌ای کاربر برای انجام کارهای روزمره مجبور نیست فرمان‌های برنامه‌نویسی را تایپ کند.

۲. دو ابزار نام‌برده‌شده در جلسه

ابزار	خصوصیت	مناسب برای
Tkinter	ماژول رابط گرافیکی همراه با بسیاری از نصب‌های معمول پایتون	آموزش، ابزارهای کوچک و برخی برنامه‌های کاربردی
PyQt / PySide	مجموعه ابزارهای گسترده بر پایه Qt	برنامه‌های دسکتاپ با رابط پیچیده‌تر و امکانات بیشتر

در سخنرانی، Tkinter به‌عنوان انتخابی ساده برای شروع و PyQt برای رابط‌های پیشرفته‌تر معرفی شد. این مقایسه باید متناسب با نیاز پروژه انجام شود؛ نمی‌توان گفت Tkinter در هیچ محصول عملیاتی قابل استفاده نیست.

۳. مثال آموزشی Tkinter

```
import tkinter as tk

root = tk.Tk()
root.title("My Python App")

label = tk.Label(root, text="Welcome!")
label.pack(padx=24, pady=16)

root.mainloop()
```

نمونه بالا یک پنجره ساده ایجاد می‌کند. در مرحله بعد می‌توانید یک دکمه اضافه کنید و آن را به تابعی متصل کنید تا با فشردن دکمه، عملیات مشخصی انجام شود.

۴. تبدیل فایل .py به فایل اجرایی

برای بسته‌بندی برنامه پایتون می‌توان از ابزارهایی مانند PyInstaller استفاده کرد. بسته به سیستم‌عامل مقصد، خروجی اجرایی تولید می‌شود. فایل exe. مخصوص ویندوز است و یک فایل پایتون صرفاً با تغییر پسوند، به برنامه اجرایی مستقل تبدیل نمی‌شود.

تمرین: با Tkinter یک فرم ساده دریافت نام و نمایش پیام بسازید. سپس درباره بسته‌بندی این برنامه برای ویندوز تحقیق کنید و مشخص کنید چه فایل‌ها و وابستگی‌هایی باید همراه آن ارائه شود.

فصل 03

معماری کلاینت-سرور و مسیر یک درخواست وب

۱. کلاینت و سرور چه هستند؟

در معماری Client–Server، کلاینت نرم‌افزاری است که درخواست را آغاز می‌کند؛ مانند مرورگر. سرور، درخواست را دریافت می‌کند و با توجه به مسیر و منطق سامانه، پاسخی مناسب برمی‌گرداند. سرور یک مفهوم نرم‌افزاری و زیرساختی است؛ در عمل ممکن است روی یک رایانه فیزیکی، ماشین مجازی یا محیط ابری اجرا شود.

۲. از ورود آدرس تا نمایش صفحه

۱. کاربر آدرس وب‌سایت را در مرورگر وارد می‌کند

برای مثال: https://example.com/products/

↓

۲. مرورگر درخواست HTTP یا HTTPS ارسال می‌کند

درخواست از مسیر شبکه به سرویس مربوط می‌رسد.

↓

۳. سرور و برنامه بک‌اند درخواست را پردازش می‌کنند

مسیریابی، احراز هویت، منطق کسب‌وکار و در صورت لزوم پایگاه داده

↓

۴. پاسخ به کلاینت بازمی‌گردد

HTML، JSON، فایل یا پیام خطا

↓

۵. مرورگر نتیجه را نمایش می‌دهد

رندر HTML/CSS و اجرای JavaScript در صورت نیاز

۳. درخواست و پاسخ HTTP

هر درخواست معمولاً دارای یک روش مانند GET یا POST، مسیر، هدر و گاهی بدنه است. پاسخ نیز کد وضعیت، هدر و محتوای پاسخ را در بر می‌گیرد. به‌عنوان نمونه، ۲۰۰ نشان‌دهنده پاسخ موفق، ۴۰۴ معمولاً نشان‌دهنده پیدانشدن منبع و ۵۰۰ نشانه خطای سمت سرور است.

۴. فرانت‌اند و بک‌اند در این معماری

فرانت‌اند بخشی است که کاربر آن را تجربه می‌کند؛ از جمله صفحه‌ها و تعاملات مرورگر. بک‌اند، منطق سمت سرور را پیاده‌سازی می‌کند: پردازش درخواست‌ها، کنترل دسترسی، تعامل با پایگاه داده، محاسبات و ارائه API. این دو بخش از طریق قراردادهایی مانند HTML یا JSON و HTTP با یکدیگر ارتباط دارند.

> **نکته:** ساده‌سازی «فرانت فقط ظاهر است و بک‌اند فقط عملیات» برای شروع یادگیری مفید است، اما در پروژه‌های واقعی فرانت‌اند نیز منطق تعاملی دارد و بک‌اند گاهی HTML کامل صفحه را تولید می‌کند.

۵. تمرین مشاهده جریان درخواست

در مرورگر، ابزار توسعه‌دهندگان (Developer Tools) را باز کنید و بخش Network را ببینید. یک صفحه را بارگذاری کنید و روش درخواست، کد وضعیت و نوع پاسخ را بررسی کنید. درباره تفاوت فایل HTML و پاسخ JSON توضیح دهید.

فصل 04

فرانت‌اند، بک‌اند، UI و UX؛ مرزها و همکاری‌ها

۱. فناوری‌های اصلی فرانت‌اند

در توسعه وب، HTML ساختار محتوای صفحه را توصیف می‌کند، CSS برای استایل و چیدمان به کار می‌رود و JavaScript رفتارهای تعاملی را پیاده‌سازی می‌کند. HTML زبان نشانه‌گذاری و CSS زبان شیوه‌نامه است؛ این دو را نباید معادل یک زبان برنامه‌نویسی عمومی در نظر گرفت.

۲. رابط کاربری (UI) چیست؟

UI یا User Interface به عناصر و طراحی رابطی اشاره دارد که کاربر با آن کار می‌کند: دکمه‌ها، فرم‌ها، رنگ‌ها، فاصله‌ها، تایپوگرافی و اجزای بصری. طراح UI می‌تواند با ابزارهایی مانند Figma طرح و اجزای رابط را آماده کند.

۳. تجربه کاربری (UX) چیست؟

UX یا User Experience تجربه کلی کاربر از انجام یک کار با محصول است؛ مانند آسان‌بودن ثبت‌نام، پیداکردن اطلاعات، فهم پیام خطا، سرعت تکمیل خرید و احساس اطمینان. طراحی UX با تحقیق کاربر، معماری اطلاعات، طراحی مسیر کاربر و آزمون کاربردپذیری ارتباط دارد.

نقش	پرسش اصلی	نمونه خروجی
طراح UX	کاربر چگونه آسان و درست به هدفش برسد؟	جریان کاربر، وایرفریم، یافته آزمون
طراح UI	رابط چه ساختار و ظاهری داشته باشد؟	کامپوننت‌ها و طرح بصری در Figma
توسعه‌دهنده فرانت‌اند	چگونه طرح به رابط واقعی، تعاملی و قابل استفاده تبدیل شود؟	HTML/CSS/JavaScript و کامپوننت‌ها
توسعه‌دهنده بک‌اند	اطلاعات، قوانین و سرویس‌ها چگونه کار کنند؟	API، مدل داده، منطق تجاری و امنیت

۴. چرا UI و UX با فرانت‌اند یکسان نیستند؟

یک طراح می‌تواند ظاهر و رفتار مطلوب محصول را در Figma پیشنهاد دهد، درحالی‌که توسعه‌دهنده فرانت‌اند آن را با کد پیاده‌سازی می‌کند. در تیم‌های کوچک ممکن است یک نفر چند نقش را بر عهده بگیرد؛ در تیم‌های بزرگ این مسئولیت‌ها غالباً تفکیک می‌شوند.

۵. مثال: طراحی دکمه ثبت‌نام

- **UX:** آیا عنوان دکمه روشن است؟ آیا کاربر قبل از کلیک اطلاعات کافی دارد؟
- **UI:** اندازه، رنگ، فاصله و حالت غیرفعال دکمه چگونه باشد؟
- **فرانت‌اند:** کلیک چگونه اعتبارسنجی فرم را آغاز کند و پاسخ را نشان دهد؟
- **بک‌اند:** داده چطور اعتبارسنجی، ثبت و از ارسال تکراری محافظت شود؟

> **نکته تکمیلی:** درباره رنگ دکمه یا جای عنصر نمی‌توان بدون شناخت مخاطب حکم قطعی داد. راه درست، توجه به دسترس‌پذیری و ارزیابی رفتار واقعی کاربران است.

فصل 05

فریم‌ورک چیست و Django چه نقشی دارد؟

۱. فریم‌ورک به زبان ساده

فریم‌ورک مجموعه‌ای از ابزارها، قراردادها و اجزای آماده است که چارچوب توسعه برنامه را فراهم می‌کند. مثال قاب آماده در جلسه برای تصویرسازی این مفهوم مطرح شد. در عمل، فریم‌ورک علاوه بر امکانات آماده، الگوهایی درباره ساختار پروژه و نحوه اجرای اجزا نیز ارائه می‌دهد.

۲. جنگو (Django) چیست؟

Django یک فریم‌ورک وب بر پایه پایتون است که برای ساخت وب‌سایت‌ها و برنامه‌های سمت سرور استفاده می‌شود. بخش‌هایی مانند مسیریابی URL، کار با پایگاه داده، فرم‌ها، مدیریت کاربران، پنل مدیریت و سازوکارهای امنیتی رایج را در اختیار توسعه‌دهنده قرار می‌دهد.

۳. اجزای کلیدی پروژه Django

جزء	وظیفه
URLconf	نگاشت مسیر درخواست به View مناسب
View	دریافت درخواست و تعیین پاسخ
Model / ORM	تعریف ساختار داده و تعامل با پایگاه داده
Template	ساخت HTML نمایشی در برنامه‌های سروررندر
Admin	رابط مدیریت داخلی برای مدل‌های ثبت‌شده
Middleware	پردازش مشترک درخواست یا پاسخ در نقاط مشخص

۴. نمونه بسیار ساده مسیر و پاسخ

در پروژه جنگو می‌توان یک تابع View نوشت که به درخواست کاربر پاسخ بدهد:

```
# views.py
from django.http import HttpResponse

def home(request):
    return HttpResponse("Hello from Django!")

# urls.py
from django.urls import path
from . import views

urlpatterns = [
    path("", views.home, name="home"),
]
```

دو بخش کد بالا نمایشی‌اند و باید در فایل‌های مربوط به اپلیکیشن جنگو قرار بگیرند؛ برای اجرا، ایجاد پروژه و ثبت URLهای اپلیکیشن در پروژه نیز لازم است.

۵. Django، Flask و انتخاب فریم‌ورک

در جلسه Django به‌عنوان چارچوب کامل‌تر و Flask به‌عنوان گزینه سبک‌تر معرفی شد. هر دو می‌توانند در پروژه‌های واقعی و محیط عملیاتی استفاده شوند. Django امکانات پیش‌فرض بیشتری دارد؛ Flask انعطاف بیشتری در انتخاب اجزا به توسعه‌دهنده می‌دهد. تصمیم درست به پیچیدگی، نیازمندی‌ها، تجربه تیم و معماری پروژه بستگی دارد.

معیار	Django	Flask
سبک کلی	چارچوب نسبتاً جامع	سبک و انتخاب‌محور
امکانات پیش‌فرض	بیشتر؛ مانند ORM و Admin	کمتر؛ معمولاً با افزونه‌ها یا ابزارهای دیگر
سناریوی مناسب	سامانه‌های داده‌محور و وب‌اپ‌های متنوع	سرویس‌ها و پروژه‌هایی با اجزای انتخابی
امکان استفاده عملیاتی	بله	بله

یادداشت تدوین: گزاره «Flask برای پروداکشن مناسب نیست» دقیق نیست. توانایی استفاده عملیاتی از هر فریم‌ورک به شیوه پیاده‌سازی، استقرار، امنیت و نگهداری وابسته است.

فصل 06

ساخت یک وب‌اپلیکیشن واقعی با Django

۱. پروژه نمونه: سامانه ثبت درخواست مشاوره

فرض کنید قرار است برای یک مجموعه آموزشی، صفحه‌ای ساخته شود که کاربر نام، راه ارتباطی و موضوع درخواست مشاوره را ثبت کند و کارشناسان بتوانند درخواست‌ها را بررسی کنند. این مثال **سناریوی آموزشی پیشنهادی** است و از متن جلسه به‌عنوان پروژه واقعی گزارش نشده است.

۲. مسئله را به اجزای کوچک تقسیم کنید

1. **رابط کاربر:** طراحی فرم و پیام موفقیت یا خطا.
2. **اعتبارسنجی:** بررسی فیلدهای ضروری و شکل داده.
3. **بک‌اند:** دریافت درخواست POST و بررسی مجوزها.
4. **مدل داده:** ذخیره اطلاعات در پایگاه داده.
5. **پنل مدیریت:** امکان فهرست‌کردن و پیگیری درخواست‌ها.
6. **امنیت:** جلوگیری از ارسال ناخواسته، محافظت از داده شخصی و مدیریت دسترسی.
7. **استقرار:** اجرا در محیط مناسب و ثبت خطاها.

۳. مسیر داده از فرم تا پایگاه داده

فرم در مرورگر ← درخواست POST ← اعتبارسنجی Django ← ذخیره Model در پایگاه داده ← پاسخ و نمایش نتیجه

۴. نمونه مدل داده در Django

```
from django.db import models

class ConsultationRequest(models.Model):
    full_name = models.CharField(max_length=120)
    email = models.EmailField()
    topic = models.CharField(max_length=180)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.full_name
```

این کد فقط ساختار داده را تعریف می‌کند. برای تکمیل سامانه به ساخت Migration، فرم، URL ،View، قالب HTML، کنترل دسترسی و تست نیاز دارید.

۵. چه چیزی یک نمونه آموزشی را به محصول قابل استفاده تبدیل می‌کند؟

- تفکیک محیط توسعه و محیط اصلی و نگهداری امن تنظیمات؛
- مدیریت خطا و ثبت رویدادهای مهم؛
- پشتیبان‌گیری از پایگاه داده و امکان بازیابی؛
- احراز هویت، مجوزدهی و محافظت از داده‌ها؛
- آزمون جریان ثبت درخواست و آزمون خطاها؛
- درنظرگرفتن حجم ترافیک و پایش عملکرد.

تمرین پروژه: ابتدا فلوچارت سامانه ثبت مشاوره را رسم کنید. سپس مدل داده، دو URL، یک View و یک فرم ایجاد کنید. در پایان با Git، سه Commit معنادار ثبت کرده و مراحل راه‌اندازی پروژه را در README بنویسید.

فصل 07

وب‌اپلیکیشن، PWA و اتصال به پروژه‌های هوش مصنوعی

۱. وب‌سایت، وب‌اپ و PWA یک چیز نیستند

«وب‌اپلیکیشن» معمولاً به یک نرم‌افزار تعاملی قابل استفاده در مرورگر گفته می‌شود؛ مانند سامانه مدیریت آموزش یا پنل مشتریان. Progressive Web App یا PWA مجموعه‌ای از قابلیت‌ها و شیوه‌های توسعه وب است که می‌تواند تجربه نصب‌پذیری و برخی ویژگی‌های نزدیک به اپلیکیشن را در شرایط سازگار فراهم کند.

۲. برای PWA چه نیازهایی مطرح است؟

- ارائه وب‌اپ روی اتصال امن HTTPS در استقرار معمول؛
- تعریف Web App Manifest برای معرفی نام، آیکون و برخی مشخصات نصب؛
- طراحی واکنش‌گرا و قابل استفاده روی موبایل؛
- در صورت نیاز، بهره‌گیری از Service Worker برای راهبردهای کش و ویژگی‌های آفلاین؛
- آزمایش تجربه نصب و قابلیت‌ها در مرورگرها و سیستم‌عامل‌های هدف.

صرفاً داشتن پروژه Django به معنی تبدیل خودکار آن به PWA نیست؛ فرانت‌اند و نحوه استقرار نیز باید نیازمندی‌های مربوط را پوشش دهند.

۳. اتصال یک مدل AI به وب‌اپ Django

یکی از کاربردهای اشاره‌شده در جلسه، ساخت وب‌اپلیکیشنی است که بخشی از عملکرد آن مبتنی بر هوش مصنوعی باشد. در چنین محصولی Django می‌تواند درخواست را از کاربر بگیرد، داده ورودی را اعتبارسنجی کند، به سرویس مدل یا کد پردازش متصل شود و نتیجه را برای نمایش برگرداند.

کاربر یک متن یا درخواست را در رابط وب وارد می‌کند

↓

فرانت‌اند درخواست را به API برنامه Django می‌فرستد

↓

بک‌اند ورودی و مجوز کاربر را بررسی می‌کند

↓

مدل AI یا سرویس بیرونی پاسخ پیشنهادی را تولید می‌کند

↓

نتیجه اعتبارسنجی، ثبت و به کاربر نمایش داده می‌شود

۴. ملاحظات قبل از راه‌اندازی

برای استفاده واقعی از هوش مصنوعی باید دقت خروجی، حفظ حریم خصوصی، هزینه فراخوانی، تأخیر پاسخ، مدیریت خطا و سیاست نگهداری داده بررسی شود. اگر خروجی مدل به تصمیم حساس منتهی می‌شود، مسیر تأیید انسانی اهمیت ویژه‌ای دارد.

ارتباط با مسیر یادگیری: یادگیری Django می‌تواند حتی برای توسعه‌دهنده علاقه‌مند به هوش مصنوعی مفید باشد؛ زیرا مدل یا عامل هوشمند، اغلب برای ارائه به کاربران نیازمند یک سرویس و رابط وب است.

فصل 08

نقشه راه تخصصی و ورود حرفه‌ای به بازار کار

۱. بعد از پایتون در یک شاخه عمیق شوید

یکی از توصیه‌های مهم خانم داوودیان در جلسه، پرهیز از تغییر پی‌درپی شاخه تخصصی بود. آشنایی مقدماتی با چند حوزه مفید است، اما برای کسب توانایی حرفه‌ای، بهتر است یک مسیر اصلی انتخاب و در آن پروژه اجرا شود. اشاره مدرس به اختصاص حدود دو سال برای قوی‌شدن در یک حوزه، یک توصیه تجربی است، نه یک پیش‌نیاز ثابت برای همه افراد.

۲. نقشه راه پیشنهادی توسعه بک‌اند پایتون

گام	موضوع یادگیری	خروجی مورد انتظار
۱	مرور پایتون و اصول OOP و خطایابی	برنامه ساختاریافته و قابل تست
۲	HTML، HTTP، مرورگر، کلاینت–سرور	توضیح جریان درخواست و پاسخ
۳	Git و GitHub و محیط مجازی	مخزن با تاریخچه تغییرات
۴	Django، URL، View، Model و Template	وب‌اپلیکیشن ساده
۵	SQL، API و احراز هویت	سرویس داده و مدیریت کاربر
۶	تست، امنیت، استقرار و پایش	پروژه قابل انتشار و نگهداری
۷	پروژه گروهی و مستندسازی	نمونه‌کار و ارائه فنی

۳. GitHub برای نشان‌دادن کیفیت کار

پروفایل GitHub می‌تواند نمونه‌ای از توانایی فنی شما باشد. صرف داشتن مخزن کافی نیست؛ مهم است که ساختار پروژه روشن، راه‌اندازی آن مستند و تغییرات قابل پیگیری باشند.

- پروژه‌ها را با توضیح مسئله، ابزارها و دستور نصب منتشر کنید.
- از فایل README، مجوز مناسب و فایل وابستگی‌ها استفاده کنید.
- کلیدهای API، رمزها و اطلاعات مشتریان را منتشر نکنید.
- کار با Branch، Pull Request و بازبینی کد را در پروژه گروهی تمرین کنید.

۴. LinkedIn و ارتباطات حرفه‌ای

در جلسه، مدرس اشاره کرد که برخی فرصت‌های همکاری بین‌المللی را از طریق LinkedIn دریافت کرده است. برای استفاده از این شبکه، معرفی تخصص، شرح پروژه‌های واقعی و تعامل مستمر اهمیت دارد.

- عنوان حرفه‌ای مشخص داشته باشید؛ مثلاً Junior Python Backend Developer.
- در بخش About توضیح دهید چه مسئله‌هایی را می‌توانید حل کنید.
- پیوند نمونه‌کار و GitHub را اضافه کنید.
- یادگیری‌ها و خروجی پروژه‌ها را بدون اغراق منتشر کنید.
- ارتباط‌گیری را حرفه‌ای، محترمانه و مرتبط با حوزه کاری انجام دهید.

۵. کار تیمی و پروژه‌های بین‌المللی

توانایی همکاری در تیم، تحویل منظم وظایف، توضیح تصمیم‌های فنی، نوشتن Issue و رعایت قراردادهای کدنویسی نقش مهمی در آمادگی شغلی دارد. حتی یک پروژه کوچک گروهی می‌تواند فرصت تمرین این مهارت‌ها باشد.

> **تمرین:** با دو یا سه نفر، یک پروژه Django کوچک تعریف کنید؛ وظایف را به Issue تبدیل کنید، برای هر ویژگی Branch جدا بسازید و قبل از Merge، یک‌بار تغییرات همدیگر را بازبینی کنید.

فصل 09

هوش مصنوعی، وایب‌کدینگ و آمادگی محیط عملیاتی

۱. نگرانی درباره آینده شغل برنامه‌نویسی

در بخش پرسش‌وپاسخ، مسئله جایگزینی برنامه‌نویسان با هوش مصنوعی مطرح شد. متن جلسه بر تفاوت ساخت یک نمونه اولیه با توسعه و نگهداری محصول بزرگ تأکید می‌کند. در عمل، ابزارهای AI می‌توانند بخش‌هایی از توسعه را تسریع کنند؛ اما نیاز به بررسی نیازمندی‌ها، کیفیت، امنیت و مسئولیت‌پذیری همچنان مطرح است.

۲. وایب‌کدینگ چه تفاوتی با مهندسی نرم‌افزار دارد؟

اصطلاح Vibe Coding معمولاً به شیوه‌ای اشاره دارد که در آن فرد با توضیح خواسته‌های خود به مدل هوش مصنوعی، مقدار زیادی از کد را تولید می‌کند. چنین روشی برای نمونه‌سازی مفید است، اما بدون شناخت معماری و ارزیابی فنی ممکن است خطا، بدهی فنی یا آسیب‌پذیری ایجاد کند.

۳. مسیر تبدیل نمونه اولیه به محصول

نمونه اولیه	نسخه آماده استفاده واقعی
فقط مسیر موفق آزمایش می‌شود	سناریوهای خطا و استفاده نامعتبر هم بررسی می‌شود
داده‌ها کوچک و آزمایشی‌اند	حریم خصوصی و کیفیت داده مهم است
اجرا روی رایانه شخصی	استقرار، لاگ، پشتیبان و پایش لازم است
کد بدون بازبینی جدی	تست، بازبینی امنیت و کنترل نسخه انجام می‌شود
یک نفر همه کارها را انجام می‌دهد	قراردادها و همکاری تیمی اهمیت دارد

۴. درباره عنوان‌های شغلی جدید با دقت صحبت کنیم

در جلسه از عنوانی نزدیک به «Vibe Coding Rescue» به‌عنوان فعالیتی برای رفع فاصله کد تولیدشده با محصول قابل استقرار یاد شد. چون این عنوان در متن به‌عنوان تجربه شنیده‌شده از یک همکار مطرح شده، نباید آن را بدون بررسی بازار، یک جایگاه شغلی رسمی و تثبیت‌شده معرفی کرد. مفهوم قابل اتکا، مهارت **بازبینی، اصلاح، تست و عملیاتی‌کردن کد تولیدشده با AI** است.

۵. نتیجه حرفه‌ای

در هر دوره فناوری ممکن است شکل وظایف تغییر کند. مسیر پایدارتر برای یادگیرنده، تقویت تفکر مسئله‌محور، توانایی بررسی کد، طراحی معماری ساده و یادگیری مستمر است؛ نه وابستگی کامل به یک ابزار یا انتظار قطعی درباره آینده مشاغل.

> **یادداشت علمی:** درباره زمان دستیابی به هوش مصنوعی عمومی (AGI) و اثر دقیق آن بر اشتغال، قطعیت وجود ندارد. پیش‌بینی‌های مطرح‌شده در سخنرانی باید به‌عنوان نظر و سناریو فهمیده شوند، نه واقعیت اثبات‌شده.

فصل 10

تمرین‌ها، چک‌لیست و واژه‌نامه

۱. پرسش‌های مرور مفاهیم

1. سه تفاوت اصلی برنامه CLI، نرم‌افزار دسکتاپ و وب‌اپلیکیشن را شرح دهید.
2. در یک درخواست وب، مرورگر و سرور چه وظایفی دارند؟
3. چرا HTML و CSS را باید از JavaScript تفکیک کرد؟
4. UX، UI و فرانت‌اند هرکدام چه خروجی‌هایی تولید می‌کنند؟
5. فریم‌ورک Django چه امکاناتی را آماده ارائه می‌کند؟
6. چه زمانی انتخاب Flask به جای Django می‌تواند قابل دفاع باشد؟
7. برای تبدیل یک وب‌اپ به PWA چه چیزهایی باید بررسی شود؟
8. چرا داشتن کد تولیدشده با AI به معنی آماده‌بودن محصول برای استقرار نیست؟

۲. مسیر تمرین پیشنهادی

تمرین A — یک برنامه خط فرمان: فهرستی از درخواست‌های کاربران را در قالب فایل CSV بخوانید و تعداد درخواست‌ها را بر اساس موضوع گزارش کنید.

تمرین B — یک رابط دسکتاپ: برای همان برنامه یک پنجره ساده با Tkinter بسازید که کاربر فایل را انتخاب کند و خلاصه‌ای از آمار را ببیند.

تمرین C — یک سرویس تحت وب: برنامه ثبت و نمایش درخواست‌ها را با Django پیاده‌سازی کنید؛ ابتدا فقط View و Template، سپس Model و پایگاه داده.

تمرین D — یک پروژه تیمی: پروژه را در GitHub منتشر و با Pull Request تکمیل کنید. برای آن README، مراحل اجرا و چند تست ساده آماده کنید.

۳. چک‌لیست آمادگی ورود به مسیر Django

- ☐ تفاوت CLI، GUI و وب‌اپ را می‌توانم توضیح دهم.
- ☐ مفاهیم پایه پایتون، تابع‌ها و مدیریت خطا را می‌دانم.

- ☐ با GET، POST و کدهای وضعیت HTTP آشنا هستم.
- ☐ نقش Model، View، URL و Template را در Django می‌شناسم.
- ☐ یک پروژه کوچک با فرم و پایگاه داده ساخته‌ام.
- ☐ با Git، Branch و Pull Request کار کرده‌ام.
- ☐ نکات پایه امنیت و احراز هویت را می‌شناسم.
- ☐ پروژه‌ام README و دستور راه‌اندازی دارد.
- ☐ مهارت‌ها و نمونه‌کارهایم در GitHub و LinkedIn قابل بررسی‌اند.

۴. واژه‌نامه کاربردی

CLI — رابط خط فرمان

تعامل با برنامه از طریق فرمان‌های متنی در ترمینال.

GUI — رابط گرافیکی

تعامل با برنامه با پنجره، دکمه و دیگر عناصر دیداری.

Client — کلاینت

برنامه‌ای که درخواست را به یک سرویس یا سرور ارسال می‌کند.

Server — سرور

سامانه پاسخ‌دهنده و پردازش‌کننده درخواست‌های ورودی.

Frontend — فرانت‌اند

بخش قابل تعامل محصول در سمت کاربر، معمولاً در مرورگر.

Backend — بک‌اند

لایه منطق، پردازش، دسترسی داده و سرویس‌های سمت سرور.

Framework — فریم‌ورک

چارچوب و قراردادهای نرم‌افزاری برای توسعه برنامه.

Django — جنگو

فریم‌ورک توسعه وب مبتنی بر پایتون.

PWA — وب‌اپلیکیشن پیش‌رونده

رویکردی برای افزودن قابلیت‌های پیشرفته وب در دستگاه‌های سازگار.

Repository — مخزن

محل نگهداری کد و تاریخچه تغییرات پروژه.

جمع‌بندی پایانی

پایتون فقط وسیله نوشتن اسکریپت نیست؛ می‌تواند نقطه شروع توسعه نرم‌افزارهای دسکتاپ، سامانه‌های وب و محصولات هوشمند باشد. برای حرکت در مسیر بک‌اند، باید معماری درخواست و پاسخ، نقش فرانت‌اند، منطق سمت سرور و امکانات Django را در کنار اصول توسعه حرفه‌ای یاد گرفت.

پیام اصلی جلسه: مسیر تخصصی خود را آگاهانه انتخاب کنید، روی یک حوزه به‌اندازه کافی تمرکز داشته باشید، پروژه واقعی بسازید و توانایی‌های فنی و کار تیمی را از طریق GitHub و LinkedIn به‌صورت معتبر نشان دهید.

خانه فناوری تهران

آموزش تخصصی پایتون، طراحی وب، هوش مصنوعی و اتوماسیون

www.fanavari.co | @fanavari.co

برگرفته از متن پیاده‌سازی‌شده وبینار، با بازنویسی آموزشی، مثال‌های پیشنهادی و یادداشت‌های تکمیلی. هیچ یک از نکات افزوده‌شده الزاماً نقل قول مستقیم از مدرس نیست.