

۱. یکی از دلایل اصلی محبوبیت پایتون، وجود کتابخانه‌ها و فریمورک‌های فراوانی است که برای پیاده‌سازی هوش مصنوعی و یادگیری ماشین طراحی شده‌اند.

این کتابخانه‌ها شامل Scikit-learn برای یادگیری ماشین، Keras و TensorFlow برای شبکه‌های عصبی، Pandas و SciPy برای محاسبات علمی، Seaborn برای تحلیل داده و NumPy استفاده از این ابزارها به توسعه‌دهندگان اجازه می‌دهد تا نیازی به نوشتگری کدهای پیچیده از صفر نداشته باشند و به سرعت نمونه‌سازی و آزمایش ایده‌ها را انجام دهند.

علاوه بر این، پایتون یک زبان مستقل از پلتفرم است و روی سیستم‌های عامل مختلفی مانند لینوکس، ویندوز و macOS پشتیبانی می‌شود، که این ویژگی باعث می‌شود کدهای نوشته شده در یک سیستم بدون نیاز به تغییرات زیاد، در سایر سیستم‌ها نیز اجرا شوند.

این انعطاف‌پذیری و پایداری، پایتون را به گزینه‌ای ایده‌آل برای پروژه‌های پیچیده هوش مصنوعی تبدیل کرده است.

همچنین، اکوسیستم پایتون به طور گسترده‌ای برای حوزه‌های مختلفی مانند پردازش زبان طبیعی (با کتابخانه‌هایی مانند NLTK و spaCy) و بینایی کامپیوتر (با OpenCV) نیز توسعه یافته است.

در نهایت، سرعت نمونه‌سازی اولیه در پایتون بسیار بالا است و می‌توان با تنها ۳۰ خط کد، مفاهیم پیچیده هوش مصنوعی را اعتبارسنجی کرد، در حالی که در زبان‌های دیگر مانند جاوا یا C++ این کار نیازمند زمان و تلاش بیشتری است.

این ویژگی‌ها باعث شده است که پایتون به یکی از کارآمدترین و محبوب‌ترین زبان‌های برنامه‌نویسی در حوزه هوش مصنوعی تبدیل شود.

۲ در بخش‌های مختلفی کاربرد دارد، مثلاً در ترمینال، در برنامه‌نویسی شیء‌گرا، و در کتابخانه‌هایی مثل `sklearn` برای یادگیری ماشین.

در زمینه‌ی هوش مصنوعی، Pipeline به فرایندی گفته می‌شود که داده‌ها مرحله به مرحله از مراحلی مانند جمع‌آوری داده، پیش‌پردازش، آموزش مدل، ارزیابی، و استقرار عبور می‌کنند. این ساختار کمک می‌کند که اجرای مراحل به صورت منظم، خودکار و تکرارپذیر انجام شود و در نهایت مدل دقیق‌تر و کارآمدتری تولید گردد.

۳. GPU‌ها (واحدهای پردازش گرافیکی) برای انجام محاسبات سنگین هوش مصنوعی حیاتی هستند، زیرا برخلاف CPU‌ها (واحدهای پردازش مرکزی) که تعداد هسته‌های کمتر ولی بسیار قدرتمندی دارند، GPU‌ها دارای هزاران هسته ساده‌تر و کوچک‌تر هستند که می‌توانند محاسبات موازی را به صورت همزمان انجام دهند. در یادگیری ماشین و بهویژه یادگیری عمیق، بسیاری از عملیات ریاضی (مثل ضرب ماتریس‌ها) قابل تقسیم به بخش‌های کوچک و مستقل‌اند، بنابراین GPU می‌تواند آن‌ها را سریع‌تر از CPU انجام دهد. به عبارت دیگر، CPU برای کارهای پیچیده و ترتیبی مناسب‌تر است، ولی GPU برای کارهای تکراری و موازی مثل آموزش مدل‌های بزرگ AI بسیار کارآمدتر است. (مثل مثالی که گفتی: CPU مثل یک فرد با دو دست قوی است، اما GPU مثل فردی است با هزاران دست معمولی که می‌تواند چندین کار مشابه را همزمان انجام دهد).

۵. به علت وجود توضیحات، بخشی از پاسخ سوال ۵ اینجاست

بهترین روش به نظرم برای یافتن عدد تصادفی، جستجوی دو دویی است، یعنی ابتدا عدد ۵۰ را وارد می‌کنیم. اگر larg باشد متوجه می‌شویم که عدد در بازه ۱ تا ۵۰ است و این گونه هر دفعه بازه حدس را به نصف می‌رسانیم. مثالی در ادامه ارائه می‌دهم که از همین روش استفاده شده و هر دفعه عدد میانی بازه قابل حدس باقی مانده را به عنوان حدس وارد کردم

The screenshot shows the PyCharm IDE interface. The top navigation bar includes 'Project' (with 'maktab HW'), 'Version control', and tabs for 'main.py' and 'Scratches and Consoles'. The 'Project' view shows a folder 'maktab HW' containing 'main.py'. The 'main.py' code is displayed in the editor:

```
# Question 2
import random

true_value = random.randint(1, 100)

while True:
    try:
        guess = int(input("Guess a number between 1 and 100: "))
        if true_value == guess:
            print(f"Congratulations, you guessed it!\nnumber is {true_value}")
            break
        else:
            print("low") if true_value > guess else print("large")
    except ValueError:
        print("Please enter a valid integer.")
```

The 'Run' tab is selected, showing the output of running the script:

```
/usr/bin/python3.13 /home/mohammad/Desktop/maktab HW/main.py
Guess a number between 1 and 100: 50
low
Guess a number between 1 and 100: 75
low
Guess a number between 1 and 100: 88
low
Guess a number between 1 and 100: 94
large
Guess a number between 1 and 100: 91
Congratulations, you guessed it!
number is 91
|
Process finished with exit code 0
```