List 3 (Final Part!)

داخل این بخش به مسائل تکمیلی از لیست و در آخر به مرور مطالب قبل میپردازیم :

جادوی لیست‌ها در پایتون!

لیست‌ها یکی از جذاب‌ترین ساختارهای داده‌ای در پایتون هستند که می‌توانیم هر چیزی را در آن‌ها قرار بدیم!

از عدد و متن گرفته تا لیست‌های دیگر، انگار یک کیف جادویی داریم که می‌توانیم هر چیزی را در آن قرار دهیم.

اما اگر بخواهیم لیست‌ها را به روش‌های **هوشمندانه‌تر** و **کوتاه‌تر** بسازیم، چه کنیم؟ اینجاست که **معرف لیست (List Comprehension)** به دادمان می‌رسد!

معرف لیست چیست؟

معرف لیست یک روش خلاصه و شیک برای ساخت لیست‌هاست. بیایید یک مثال ساده بزنیم:

روش قدیمی و خسته‌کننده:

```
sq = []
for x in range(10):
    sq.append(x * x)

print(sq)
```

خب، این کد کار می‌کند، اما چرا وقتی می‌توانیم **باهوش‌تر** بنویسیم، خودمون را اذیت کنیم؟!

روش جدید و حرفه‌ای با معرف لیست:

```
sq = [x * x for x in range(10)]
print(sq)
```

دیدید؟ هم کوتاه‌تر، هم خواناتر، و هم باعث می‌شود دوستان پایتونیست‌تون به شما حسادت کنند!

چند مثال جالب از معرف لیست

یک لیست پر از صفر بسازیم! (چرا؟ چون می‌توانیم!)

```
zeros = [0 for _ in range(5)]
print(zeros)
```

ترکیب اعداد و حروف (مثلاً برای ساختن نام کاربری تصادفی!)

```
usernames = [str(num) + letter for num in [1, 2, 3] for letter in ['A', '
print(usernames)
```

فیلتر کردن لیست (مثلاً فقط عددهای مثبت را نگه داریم!)

```
data = [10, -4, 53, 122, 0]
positive_numbers = [x for x in data if x > 0]
print(positive_numbers)
```

توان دوم همه اعداد یک لیست را حساب کنیم

```
squared = [x * x for x in data]
print(squared)
```

اشتباهی که همه مرتکب می‌شوند!

```
[x * x, x for x in data]
# SyntaxError: invalid syntax
```

چرا خطا داد؟ چون در معرف لیست فقط **یک عبارت قبل از** `for` می‌توان نوشت!

لیست‌های تو در تو (Nested Lists) - وقتی لیست درون لیست باشد!

لیست‌ها می‌توانند درون هم قرار بگیرند، درست مثل جعبه‌های داخل یکدیگر! این کار خیلی به درد **ساخت آرایه‌های چندبعدی** می‌خورد.

یک ماتریس ۳×۳ بسازیم:

```
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

for row in matrix:
    print(row)
```

یک آرایه `n × m` پر از صفر بسازیم (به سبک حرفه‌ای‌ها!)

```
n, m = 3, 4
zeros = [[0 for _ in range(m)] for _ in range(n)]
print(zeros)
```

حذف کردن آیتم‌های لیست (وقتی از چیزی خسته شدیم!)

گاهی اوقات نیاز داریم **چیزی را از لیست حذف کنیم**، این کار را می‌توان با `del` انجام داد.

مثال: لیست جدایی‌ها!

```
a = ['TA', 'rafte', 'key', 'barmigarde?']
del a[0]
print(a)

del a[0:2]
print(a)
```

پاک کردن کل لیست (وقتی از دستش خسته شدیم!)

```
del a[:]
print(a)
[] :خروجی #
```

حذف کامل یک متغیر (انگار که هرگز وجود نداشته!)

```
name = "salam"
del name
print(name)  # NameError: name 'name' is not defined
```

چاپ یک‌خطی لیست (وقتی حوصله `for` زدن نداریم!)

پایتون به ما اجازه می‌دهد که لیست را با یک ترفند **خیلی سریع و زیبا** در یک خط چاپ کنیم.

بدون نیاز به حلقه:

```
ls = [0, 1, 2, 3, 4]
print(*ls)
```

پس : خیلی راحت! `*ls` یعنی تمام عناصر لیست را بردار و جدا جدا چاپ کن.

جمع‌بندی: چرا معرف لیست؟

کد کوتاه‌تر و **خواناتر** می‌شود.

حلقه‌های اضافی حذف می‌شوند.

عملکرد بهتری دارد.

**باعث می‌شود حس کنیم خیلی حرفه‌ای هستیم! **

** حالا نوبت شماست! آیا تا به حال از معرف لیست استفاده کرده‌اید؟ یا همچنان طرفدار روش‌های کلاسیک هستید؟!**

جمع‌بندی کلی:

ما توی این درسنامه با هم یاد گرفتیم که:

- **متدهای لیست** مثل `append()` ، `pop()` ، `clear()` و ... چطور می‌تونن لیست‌ها رو راحت‌تر مدیریت کنن.
- چطور با استفاده از **معرف لیست** کدهای پیچیده رو ساده و خوشگل کنیم.
- چطور **لیست‌های تو در تو** بسازیم و برای کارهای پیچیده‌تر از اون‌ها استفاده کنیم.
- با **دستور del** می‌تونیم هر چیزی که نمی‌خواهیم رو از لیست یا حتی متغیرها پاک کنیم.
- چطور لیست‌ها رو با استفاده از `*` توی یک خط چاپ کنیم.

پس، پایتون رو دست کم نگیر! این زبان کم حجم و قدرتمند می‌تونه دنیا رو برای شما جذاب‌تر کنه.