Key takeaways: don't stop worrying, but love atomic()

Don’t use floats.
Don’t use round(), if you must, always account for the remainder.
Don’t execute non thread-safe writes in a parallel environment.

Use `Decimal` instead of `float`, and `Decimal.quantize()` instead of `round()`:

```
Decimal('0.35') + Decimal('100.15')
```

Lock dependent rows during transactions:

```
with transaction.atomic():
    players = Player.objects.filter(user=user).select_for_update()
    user.balance = player_balance_sum(players)
    user.save()
```

Use atomic compare-and-swap operations when you cant lock:

```
User.objects.filter(id=user.id, balance__gt=50)\
            .update(balance=F('balance) - 50)`
```

Q&A

Special thanks to:

Django Core Team & Contributors,
PyGotham Organizers, Andrew Godwin,
Aphyr, Tyler Neely

Final Disclaimer: I'm not qualified to tell you how to
design your distributed system. Get a professional for that.

I can only show you challenges and solutions I've discovered in my personal adventures with Django. Please let me know if you have corrections!

Monadical is hiring remote developers!

Ping me @theSquashSH on Twitter!