Thank you for having me!

No, really: it's hard to overstate how much getting to be here means to me. This is literally the highlight of my professional career - here, right now.

This is my *eleventh* PyCon! Since I first came to PyCon I've gotten married, moved across the country twice, bought and sold two houses, a couple of horses, and dozens of chickens... This talk, in fact, is on the tenth anniversary of my first PyCon talk -- I first spoke in 2005 in DC. That year, Adrian demoed some tools we'd been using to build websites at the Lawrence Journal-World - a thing we then just called "The CMS", which y'all now know as Django. I've watched this conference grow from 300 people to 3000, watched the community explode with new faces and growing diversity.

So when I say this is the highlight of my career, I'm not exaggerating - this conference, this community, is the one that means the most to me, and to be up here is... special. Really special.

Unfortunately, though, the main thing I'm feeling right now isn't pride; it's insecurity. There's a little voice in the back of my head telling me: "you don't deserve this - what are *you* doing here?" From the moment I was invited to speak, this voice was in my head. Jay Smooth calls this voice "the little hater":

(movie clip)

The really insidious thing about The Little Hater is that he's a little bit right. It's not that I haven't done work that I'm proud of, but they're probably not the things most of you are thinking about.

See, most people think of me as "The Django Guy" - one of the "creators" of Django, a "BDFL", etc. Most of you probably assume that I'm speaking to you because I'm an amazing programmer - a "rock star", a "ninja" or whatever.

Not true. I am, in fact, a quite mediocre programmer.

This is [Ann Trason](http://en.wikipedia.org/wiki/Ann_Trason). She's perhaps the most accomplished ultrarunner of all time: she won the Western States 100 -- a 100 mile race from Squaw Valley to Auburn -- an astonishing *14* times. She set dozens of course records in the '80s and '90s, many of which still stand today. In 1994 she set the course record at the Leadville 100 -- a 100 mile race which takes place mostly above 10,000 feet, including a climb over a 13,000 ft pass. Since then, only a small handful of runners have come within an hour of

breaking her record. It's never even been close. It's hard to overestimate how incredibly dominant she was; for nearly two decades, she won nearly every race she entered.

As of a few weeks ago, I'm an ultrarunner too - I finished a 50k, which means I get to, nominally at least, label myself with that same "ultrarunner" label that Trason wears, too. I am, of course, not even remotely in her league: I'm a solidly mediocre runner -- at my 50k, I finished 535th out of about 1,000. I'm average, and I have numbers to prove it!

We can fairly objectively measure the difference between Ann and me. We might quibble over what technique to use -- Do we compare time? Pace? How do we account for difficulty of a race? -- but we've got plenty of numbers, objective metrics to compare our performance. There's even a site, UltraSignup, that calculates what they call a "runner score". Mine's 68% which means that, across my races, I typically finish in about 68% of the time of the winner. Ann's is 99%, which means she finishes, on average, in 99% of the time of the winner -- in other words, she usually wins.

It isn't surprising that there's someone that much better than me. Running performance breaks down into a pretty bell curve. Most people are average, and a few people are spectacularly above or below average. This isn't much of a surprise: when we look at most kinds of performance, they tend to show a results like this. This pattern is so common that we even call it a "normal" distribution - it's so common we consider it "natural", almost a law of nature. Nearly any skill that we know how to measure shows this kind of distribution.

When I said a few minutes ago that I was a mediocre programmer, some of you probably didn't believe me. But why? We know that skill tends to follow a normal distribution, which means that if we assume that someone falls into the fat part of the curve we'll be right, most of the time. So why'd you assume that I'm under the skinny end?

We don't know how to measure programming ability in any remotely systematic way. So we make up stories to fill that gap. The stories we make up are simple: that person "sucks" and programming. This other person "rocks". We seem to fall into an implicit assumption that most people are either awesome or terrible. So because I'm up here, because I'm associated with Django, you pick one of those stories. The "sucks" story doesn't fit me, so you assume that I must "rock".

But we know from studying all sorts of skill that if we *could* measure programming ability, most people would be average. Because most people are average at most things!

This assumption that programming skill falls into this weird "sucks-rocks" distribution is pernicious, but it's a myth. This "talent myth" sets up a world where you can't program unless

you're not a "rock star" or "10x programmer". This couldn't be further from the truth: most of us are mediocre programmers, and that's OK! I'm here to defend mediocrity.

By 2020, the Bureau of Labor Statistics expects there to be a 1.5 million job shortfall in the tech industry. The EU has published similar statistics. It's clear we need to get more people doing what we do. We call this sometimes about a "pipeline problem" -- the problem of getting more people into tech. The talent myth stands in the way here, preventing otherwise-interested people from learning technology.

If we believe that programming talent is distributed bimodally, this leads to a belief that programming ability is something "innate", a talent you either have or don't have. If the only options available to programmers are to be amazing or terrible, we believe that you have to be "passionate" about programming: it has to be more than a job, more than a career; you need to be spending most waking moments thinking about software.

We don't believe these things about other kinds of activities. Last year, over half a million people ran marathons. Did all of them have "innate" talent? Of course not; most of them, like me, did it rather slowly. Nor do we believe you have to be "passionate" about running; in fact, many people who run actually kind of hate it.

A few years ago, I attended KU's GIS day. One student gave a particularly incredible presentation about mapping and predicting flood data. The tools she used will be familiar to many of the people here: AWS, Linux, PostgreSQL, Python, Django, etc. I was hiring Python developers, so I approached her after the talk and asked if she was interested in interview with us; she replied, "oh, I don't think so, I'm not really a programmer." This from a woman who'd just written thousands of lines of Python and had invented her own distributed GIS data processing pipeline.

We need to start to recognize that programming isn't a passion or a career or even a job -- it's just a skill. Actually, it’s even more complex than that; successful software development requires all *kinds* of skills: writing, communication, design, project management, debugging, understanding unicode, and on and on. There are multiple, independent skills which go into “being a programmer.” When we assume all these parts of inextricably linked to another, it breeds an assumption that a person is the min() of their parts.

Like any skills, you can do these things professionally, or occasionally; as your full-time job, as part of it, or as a hobby. You can do them well or badly or -- most likely -- you can be average. If we embrace the idea that it's fine to be simply OK at programming, it'll be less intimidating when beginners dip their toe in.

Past these entry barriers, the talent myth haunts even established programmers throughout their careers. This is difficult to talk about, but important to acknowledge: anyone who's watched tech closely knows that our industry is rife with sexism, racism, homophobia, and other forms of discrimination. This problem is complex and multifaceted, and the talent myth is one of those facets.

In the industry, the talent myth gets recast as the myth of the "brilliant asshole" - the "10x programmer" who's an incredible talent, but completely terrible to other people. People often defense these jerks -- they're, the argument goes, that we have to "put up with" their toxic behavior. In reality, we know that skill is normally distributed, so it's unlikely that these people are really that much above average, if at all. Even if they are: how many people does "10x" programmer get to drive out of tech before it's a wash?

More insidiously, this idea that programmers fall neatly into a "sucks" and "rocks" bucket biases us dangerously. Quick: think of a "rock-star programmer" -- what's the image in your mind? Most of us are probably picturing a young, white man, because that's who've we've seen in the media, on tech company mastheads, and so forth. This means, of course, when we see someone that doesn't fit that stereotype, we're prone to assume they're not a "real" programmer. Almost all the women programmers I've talked to have a story about someone assuming they're not a programmer. Here at PyCon, I've talked to multiple women who have been asked what guy they're "here with".

On the other hand, when because I look like a "real" programmer, people assume that I am a fantastic developer, when the truth is I'm just... OK.

These sorts of toxic assumptions about what a "real" programmer looks like contribute to the poor attrition rates of minorities in tech. For example ([cite](http://www.ncwit.org/sites/default/files/legacy/pdf/NCWIT_TheFacts_rev2010.pdf)), women leave the tech industry at twice the rate of men. 41% percent of women leave tech within ten years, compared to 17% of men. There are certainly other causes, of course, but imagine how frustrating it must be to have decade of experience, and still have people assume you're terrible at your job just because you're a woman.

It's going to take serious effort to fix tech's diversity problem. We're never going to get there without figuring out a more nuanced way to think about programming skill and ability. There are all kinds of runners: sprinters and distance runners and marathoners and ultramarathoners, trail and road runners, professionals and amateurs, and runners of all sizes, shapes, genders, ages, and races. All of them have different metrics for success, are capable of being successful on their own terms. We have *got* to find such a nuanced, fair understanding of skill in tech.

I want to close with a little story. This talk begin with a conversation that I had at PyCon some years ago, a conversation that has stuck with me. It planted the germ of an idea in my brain, and that idea grew into this talk.

I was talking with Lynn Root, an engineer, the founder of the San Francisco chapter of PyLadies, a PSF board member, and last night's auctioneer. I was excited about the people that PyLadies was bringing into tech, and I said something to the effect of, "it's just so great to see so many badass women programmers".

Lynn said, "yes, but it's a shame that they have to be bad-ass. We'll know when we've *really* been successful when we have a bunch of mediocre women programmers."

It's true: the myths we tell about talent set an implausibly high bar for entry. We need to dismantle these myths, and build a community where we recognize that average is... pretty great!

Hi, I'm Jacob, and I'm a mediocre programmer. Join me, and let's all do some perfectly average programming!

Many thanks to Lynn Root, Alex Gaynor, Glyph, Leigh Honeywell, and Noah Kantrowitz for their inspiration, feedback, suggestions and comments.