Djangocon2015.

Sunday 31st May 2015

Cardiff University, Main Building

DANIELE PROCIDA: Good morning, welcome to Cardiff and welcome to Djangocon Europe 2015 our open day so I hope there are a few people at least who are not part of the main conference but have come just for today. Oh great. 2 people twice as worthwhile as 3. Put your hand up if you are here just for the opening. That's great wonderful. {applause}.

Unfortunately slightly bad timing because it's half term and middle of exams so I don't think we'll get all the sixth-formers who might have been here otherwise. Obviously it's quite early on a Sunday morning for sixth-formers and because of the debacle we had with one direction we had to change our dates a bit so that hasn't held with the timing for school pupils but any way ...

First of all, if you need help, just ask a volunteer and a volunteer is wearing a blue lanyard. Any Committee Members in here could you stand up - there is Vincent and David. Any volunteers apart from the Committee? There is Ben. So you'll see a few of us wearing these and just ask us for anything and if we know the answer we will help you.

Speaking of lanyards at this conference we have a photography policy so if someone is wearing a black lanyard which is most people that mean they're happy to be photographed. If they're wearing a white lanyard they don't wish to be photographed, they don't wish to be asked if they're sure and they don't wish to be asked why they don't want to be photographed. People have various reasons so please respect this policy. If someone is wearing a blue lanyard I think they want to be photographed as much as possible!

If you are an open day visitor and don't have a lanyard but don't want to be photographed you can borrow a white one for the day.

As every Python or Django conference we've governed by a code of conduct and the code of conduct is there to ensure everyone who attends is there to enjoy the event, that everyone feels welcome and feels like they'd like to come back to this again. If you've got a programme booklet it's in full in there. If you want to look at it, have a look at our web-site it's one of the most prominent things on there.

In brief, it is that nobody at the event is to suffer harassment or abuse of any kind and no one should behave in a way that causes distress to anyone else. If you become aware of a problem or something bother S you please come and speak to a volunteer immediately and we have a written procedure for dealing with it. Not that we expect anything to happen, partly they don't happen because we have these policies.

So, what is a Django con?

It's an open source software community conference. It's a non-profit conference run by volunteers. When I say volunteers, even we the members of the Committee are unpaid volunteers. In fact everybody who comes to Djangocon including the members of the Committee who organise it have to buy their own tickets. That's the way it works that everybody pays to attend. The only people who don't pay to attend are people who need financial assistance and can apply for financial help for a free ticket or even for things like transport and accommodation. So, bear that in mind. It might be different from other software conferences you've been to. But it is important and it's why it feels more like a festival or a party than a conference and why the first thing people do when they turn up and ask how can I help. Yes a bunch of attendees asking how they could help by volunteering so it's a nice thing to be involved with.

So thank you to all our ad hoc volunteers who have volunteered.

This is our programme for our 6 days. It's our open day which will give you a taste of it. Monday to Wednesday we'll have our formal programme of talks not at the university but at Cardiff City Hall. Then on Thursday and Friday we'll be back here at the university for 2 more days of workshops, clinics and sprints.

If you want to look at the web-site it's at Djangocon.eu. Timetable is on the web-site in your programme booklet and is up on the walls in various places and we'll get more copies printed as we speak.

We've got various workshops there is Django Girls all day. This is another non-profit organisation that has been set up to help encourage more women into computing and software development. It's been extraordinarily successful, it's been running less than a year and already has held workshops in I'm not sure how many dozen countries.

During the day we've got various workshops in different places starting from 11 o'clock. Very briefly we have Python and Django for PHP coders not because we're trying to poach anyone.

NEW SPEAKER: Yes we are!

DANIELE PROCIDA: Yes we are ... Tracey Osborn, not sure if she is here but she's doing a Django tutorial specially aimed at people with a background in web design but not in programming.

Stefan is doing Django CMS. An introduction if you are interested in web content management. Marcus is doing a tutorial for people who have some Python experience already but would like to start applying it to the web.

Leonardo is offering a workshop to dive into object oriented Python for people with some Python but want to understand more. And Erik is doing a tutorial on cryptography which will be suitable for people who only have a minimal grasp of Python and mathematics.

Go to them and have fun. You should have registered for them if you want to go but if you haven't registered ask nicely at the door and there might be a place for you.

The talks are in batches, 2 before lunch and 2 after lunch. We'll try and compress the time that we have lost a bit. We have breaks at 11 o'clock, lunch will be served from 12.30 to 14.30 at Aberdare hall there will be signs for people to take you there, and there will be an afternoon break at 4 o'clock. And we aim to finish by about half-past 5.

This only applies to the people who signed up for the whole conference. Please, you must tell us which meals and dinners you are planning to come to. So if you haven't you need to come and see somebody in the registration desk as soon as possible otherwise there may not be dinners for you. This is part of our policy on food waste. We don't want to be catering for people who don't turn up for meals so we only cater for people who have told us they are coming to meals. That includes the main conference dinner tomorrow night if you haven't told us that you're coming for that it means you do not have a ticket and if you do want to come and don't have a ticket it means you will have to go on a waiting list so come and see us about that as soon as possible. If there is a mean you signed up for but you're not coming to please either tell us or remove it from your ticket on-line so we can allocate it to someone else. We'd be very grateful.

Also on the topic of waste you'll notice that everyone who registered for the full conference has a bottle - can you wave that bottle - so we're not serving, not using plastic cups or any other plastic bottles. There are water coolers. Fill up your bottle. That bottle will not last for ever but as long as you do if you look after it.

Our sponsors are setting up in the Jones gallery in the foyer. Go and talk to them. They're as much part of this community as everyone else is. Some are here just for the hell of it because they want to participate in the community. Drop by, say hello, they'll appreciate you talking to them and there will be a jobs fair so they'll be expecting to see you, a lot of them are recruiting at the moment, drop by between 12 and 2. They're making a huge contribution to the conference and we're very grateful for that, couldn't possibly happen without the sponsors, would be about 2 or 3 times the price if it weren't for our sponsors.

Finally there is Cardiff university that's a really important part of this conference and we're very grateful for the contributions they've made.

Several members of the committee are from Cardiff University. Most of the volunteers are students of Cardiff University and they've provided facilities resources and the staff so thank you for giving up your Sunday for us {applause}.

The university through the Vice Chancellors Office has provided a number of scholarships for Cardiff University students to attend conference. A number of the schools have offered funding to their own students separately or given tickets as prizes to their students so we've got a lot of Cardiff University involvement. Also got some speakers 3 of whom you'll be hearing this morning and we also have 3 very special visitors who have travelled from Namibia and that is also partly in thanks to Cardiff University so welcome to Maria, Michael and Jessica all the way from ... {applause}.

If you are functioning on twitter do please mention and thank Cardiff University because they have made a huge contribution to this - or whatever social media....

That's enough from me. I am going to introduce Prof. Roger Whitaker who is Dean research in the College of Physical and Engineering Sciences. Physical sciences engineering. And do you need a - no...

PROF WHITAKER: Just going to say a few words.

DANIELE PROCIDA: Thank you Roger. {Applause}.

PROF WHITAKER: Good morning everybody and thank you for the invitation to come along. I'm Roger Whitaker Dean of research for physical sciences and engineering and I'm also here in a personal professional capacity because I'm a researcher in computer science and in particular mobile and social computing and the group that I work with have made really good use of Django over the years so it's something we like to see prosper. But on behalf of Cardiff university and on behalf of the Vice Chancellor it's a really great pleasure to welcome again the event at Cardiff and it's nice to see so many people here, it requires dedication to come along, as a delegate on a Sunday morning when it's raining, so very impressed indeed.

I'm biased because I'm in the discipline but Django is really important and I was asked by some senior staff in the university what Django was and I managed to recall the tag line around helping perfectionists meet their deadlines and I think that sums it up really well.

I think also what particularly pleasing about this event and much of the activity going on at Cardiff in this area is that it is very much grassroots participation and snow balling of people's interest collectively that has led to so much activity in this area. And that goes hand in hand with what the university is trying to achieve and is much higher on the university agenda at the moment and the umbrella term is innovation and innovation in whatever form is something the university is keen to help try and catalogue and I just thought I would say to close, I know we're running behind so I just want to keep this short and sweet, but I just thought I'd give a few pointers around some of the investments that the university is making in terms of innovation in ways that the university here at Cardiff hasn't done so in the past, this is very much new activity.

First and foremost, there is a very large project going on in the order of £200 million and innovation system which is an extension to our campus on the Maindy Road site and there are a number of facilities and centres that have been earmarked and developed to go forward around the area of innovation.

One of them on that site is the university social science park which will be one of the first kinds of facilities certainly in the UK and beyond and within that there will be a computational social science lab where people from different disciplines will be interacting with ways that we haven't seen before.

There is also going to be an innovation centre there on site, a separate facility, which will allow SMEs and companies to come in and work alongside Cardiff university staff, Cardiff university research.

The university is also planning data innovation institute which is going to be a large pan university facility that will be working across biological and life sciences, arts, humanity social sciences and also physical sciences and engineering bringing together people around data science and broadly speaking pulling knowledge from data in new ways. So these are really exciting times for innovation at Cardiff not least in the technological domain.

Last but not least I should mention my home school computer science and informatics which is developing a new way of teaching software engineering so that is going to be coming forward and advertised very shortly but something that is intimately connected to Django and this conference and I am sure there will be a way to link those activities together.

So, on behalf of the university I hope it is a great success, I am sure it will be and thanks to everybody involved organisation.

{Applause}.

By the way, if you are just an Open Day visitor and don't have a lanyard but would like one of these lovely, member of the Djangocon Community badges, go the Council Chamber and ask for one, then you will have an official badge.

Okay, very pleased to welcome Russell Keith-Magee who is the President of the Django Software Foundation, a job glamorous as it sounds, travelled from Perth in Australia, I will let him say all the rest thank you Russell.

(APPLAUSE).

RUSSELL KEITH-MAGEE: Thank you. Well good morning everyone as I am Russell Keith-Magee, I would like to ask you a question, what do you see when you look at a computer? A lot of people look at a computer and what they see is a fancy typewriter or a fancy calculator or a fancy postman delivering your electronic messages.

What I see?

I see Lego, now a typewriter, a calculator, these are tools when the postman comes with the parcel and doesn't ring the doorbell. ... These tools are single purpose devices they do one thing, a computer is also a tool, like Lego, infinitely recon fig rabble, just as Lego can be any toy you want it to be, a computer is a tool you want it to be. You want to have an idea of what you want to do. If you need a typewriter you can make one.

If you need a calculator, you can make one.

If you need someone to deliver your mail, it can do that as well.

Anything you can conceive of, or you can programme a computer to do it, the catch of course is that in order to make any of your dreams into reality, you have to be able to programme the computer to realise your dream.

Over the years, two things have changed, if first is the size of computers the first ones were big, filled an entire room. These days almost everyone who is in this lecture hall can be carrying in their pocket a computer considered a supercomputer decades ago.

But, secondly, the way you programme computers has changed a lot ore the years as year, this was ... patch chords, you ran wires through, you can see the ladies doing.

The configuration of the chords can control the calculation that was being made, after a while, people worked out how the replace the patch chords, but then you are working close with the metal. You are working with low level instructions.

This is Grace Hopper, in this picture, holding the piece of magic, the Cobol programming manual, a machine independent programming language, the idea to express the solution to a complex problem using a high level abstract language...

Interestingly, this actually exploit it is strength of computer to make them more powerful, computers are really good at taking a set of instructions and following the rules over and over again. Converts from a high level to a machine language is a matter of following a set of rules over and over and again.

As a result you now have abstracted higher level way of communicating what it is you want to computer to do.

This gains you two things, first the concepts you can express that you are able to express easily, become more complex.

If you want to bake an apple pie from scratch, first create the universe, early computing was like that. You had to make everything from scratch.

High level language, when you bake the pie, assume flour, ... the less time you have spent time bogged down in flour, the more time you can spend making pies.

The second thing is you make mistakes they are a lot easier to find, in computing circles those are often refer to as bugs, why? Because in the early day they were bugs, this is Grace hopper's logbook, a moth stuck in a mechanism, causing the calculations to go haywire.

So these newfound powers to ability to encode complex ideas, the ability to fix problems, computers started to be used to do more interesting things, twenty years ahead, this lady, Margaret Hamilton. The reason that Neil Armstrong walked on the moon.

In this picture, standing next to the code for the lunar landing, if you listen to the transcripts, the last thirty seconds, you can hear him say, programme alarm code ... that is an error condition being reported by the computer on board the lunar lander, warning, too many requests passed to the computer and certain operations ignored as a result.

Now the reason that that error message wasn't a problem, programmed the computer to prioritise important over other operations, one of the reasons they got it right, using a high level programming language that allowed them to express the priorities in the case of an overload.

Okay, wind toward another twenty years, Commodore 64. Commodore 64. Gave me a blinking prompt, started in basic, it was called that for a reason, an approachable language compared to other that is existed at the time.

Of course that didn't mean it was great, but just better than the alternatives F you wanted to do anything interesting had to understand all sorts of internal details, drawing on the screen, using operations to write into the memory of the machine I was able to programme, it could be done. Now a lot of time that was just meant typing in code I found in the back pages of computer magazines. It was coding, my first game, sand castle type the number appeared on the screen. The length of time, every time you typed in the number, a bit of sand thrown on to the sand castle, the tide was coming in, eventually the tide wash away your sand castle. Grand Theft Auto, eat your heart out!

To me as a nine year old, I turned this into a new source of fun, it could do something tangible, that an hour before it couldn't do.

So wind forward another twenty years, so what do our tools look like today? The languages continue to be more and more expressive, the first part of the talk, Python. It isn't named after a snake; it is a high level programming language, not only does it abstracts away the hardware, but the operating system.

Where BASIC likes to call itself a beginner’s all-purpose language, Python has a much better claim to the crown, descended from language, abc, designed as a teaching language to replace basic. Python replaces a lot of the properties, but extended the ideas into a powerful all-purpose language.

Now one of the reasons that it is a good teaching language, the underlying philosophy, aimed to be straightforward and not too clever, it has a set of unofficial guiding principles referred to as the ZEN of Python.

[on screen].

Now, hopefully, it is clear why these are all good properties for a language that you are going to teach someone to programme, explicit you always know what is going on, simplicity means it is easy to understand what is going on, readability and beauty, make it easy to consume whatever it is you are reading.

But as it turns out, these principles make for expert programming language as well. A running joke that you are only half as smart when you are debugging code as when you wrote it.

Having a language that encourages you to be explicit about what you are doing encourages you to embrace your ability ... these are properties that make it easier to read your code in a year's time, but for someone else in the team, decipher what it is that you have done.

Just because Python is clear and legible doesn't mean it can't be powerful. It has really strong housing introspection, code that is running; work out what is running, why it is running and change the way it is running as a result. An object can look at another object and say, does that have property X and respond appropriately. Means you can Meta programme, write programmes that write programmes.

... the further you can abstract the more power you get, the more power you gain and the more specialised you can become. Plus, programmes that write programmes are the happiest programmes in the World.

Python comes out of the box with a library of built in tools, maths, dates and times, data compression if you have all you have got a standard install, you have got a huge amount of power.

Python community, congregates ... known as the Cheese Shop, the Cheese Shop is a massive library, everything they couldn't put into the box ... it is a huge range of packages for every conceivable purpose, reading every possible file time and interfacing with any possible system you can think of.

So when faced with the problem of exploiting a wonderful international telecommunication networks we call the internet, inevitable someone will come up with a Python package to help you to build in a website.

That is where Django comes in.

What is it? It is a high level Python web framework that encourages rapid development and clean pragmatic design.

A web framework is a library of software that abstracts out the common development, short cuts for tasks that you have when building a website, dealing with log ins and maintaining a session state. Permissions to make sure that the right people can see the content you need.

A good web frame work finds the pain points and smooths them over but not getting in the way, working at the high level abstraction, so you don't need to worry about the protocol or cookies or how the database is working, shouldn't get in your way if you want to dig into the internals to make it work.

Django, it builds on top of what Python does. If you are using Django, you can use Python.

So, it encourages rapid development, regardless of how many powerful features a language has or a framework has, a web framework is worthless if it doesn't save you time ... with Django, build websites in a matter of hours not days.

This comes out of a set of real world programmers.

When a big story broke they didn't have the luxury of a long development cycle, the tools are there in Django to make you more productive, to help you get your grand idea from your head into the world as quickly as possible.

Encourages clean pragmatic design, maintains a, tries to maintain a clean, desire through its own code and makes it easy to follow best practices in the applications you are creating. The philosophy to make it easier to do the right thing.

For example, there is a group out there on the internet, open web application security project. Who its name suggests, draws attention to the security in applications, every couple of years, publishes the top ten security issues.

One of the reason the organisation exists, many of the tools outed there for developing web software don't have a good security story, sure you can build a secure website in php, it can be done ... if you learn php by reading tutorials on the web, the chances are you have unwittingly embedded some security problems in the website, slow the pick up on the fault, having security as a default, having something difficult to do something the wrong way, ... in Django, it is the default setting, it is (INAUDIBLE).

But, if you follow the path of this resistance, if you do what the documentation is telling you to do, you at least won't introduce security problems accidentally. As the professor said before, the (INAUDIBLE).

Like I said, Django came from the pressures of the news room, to the credit, news rooms that it came out of, the small newspaper in Kansas, they didn't think of the website as the way of printing newsprint on screens, they introduced the idea of data journalism, it is presenting the information to the reader in a way that is compelling.

So you don't just write a thousand words saying the crime is on the increase, you show a map, showing you where the crimes occurred, trends over time, break them down by crime time, interesting highlights but let the user explore the data for themselves, given access creates content and perspective to the argument you are trying to make, allow it is user to draw their own conclusions, what is compelling to you, may not be to someone else the things that convinced you of the argument, won't be the connection that convinces someone else that the argument you are putting forward is correct.

For me, that is why it is worthwhile learning Django, some much of our lives are governed by data, logs, use of a resource, ... if you look at your computer as a fancy note pad, you might be inclined to take the data, stick it in a word document and be done with it.

But that is hardly a good use of that data. I am constantly amazed how many businesses I encounter, where the core is the "spread sheet" the master list of things can be done, how it is managed. This document can be used by one person, the master of the spread sheet. Nothing gets done unless you talk to the person, master of the spread sheet.

My day job...

I use Django all day every day to solve this exact problem, at its core, (INAUDIBLE) is a business, we ... a customer has a leaking pipe, every customer has an address, the business owner needs to know what is outstanding and what needs to be done.

Employees need to know ..., how much the customer is invoiced for. When I go to the company, the admin strategy, one of two things, a huge pile of printed work orders this deep, of everything they need to have done or "the spread sheet" which lists all the jobs to with done.

We are able to replace it with a website that anyone in the company can look at and anyone in the company can update. Everyone has a full history of everything that is done on the website, they can see it on the website, or sitting on the beach in Bali.

The more you look through the lens of data, the more you see day ... to be visualised in a more engaging way and while in the past turning the dreams into a reality, might be a pipe dream., with ... you have the tools to make the vision a reality if you have, if you have got a problem ... easy to start a business, the website, rent servers by the hours, pay tens of dollars a month.

But you don't want to start your software empire, you don't have to publish the...

If you need to know how to turn the bricks together, you can turn the computer into whatever you want it to be.

Python and Django, all open-source projects, doesn't anything to down load them, when you get them, you get all the source code as well. Lego kit, a set of instructions, most people follow the instructions, if you are curious, you can pull it apart, see how it works, connect it to another model.

If something breaks you can fix it yourself, if you find something interest you can share the knowledge.

Open-source...

If you find a problem ... (INAUDIBLE) ... not only can you add the new feature, you are encouraged to add the new feature and...

So that is my pitch for why you should learn Django and Python. I hope that has teased your interest. ... I hope you have a great day and I hope I can see you around the community very soon. (APPLAUSE).

DANIELE PROCIDA: Thank you very much Russell, you'll see and hear a lot more of Russell during the week and - no don't try and avoid him.

NEW SPEAKER: Our next speaker is Mark - {inaudible} he's going to be speaking about web framework for the creative mind. Whenever you are ready Mark and as you are running out of time I'll be putting up pointers.

MARK STEADMAN: I am timing myself any way and the time starts now. Hello. My name is Mark. I wanted to talk about making the case for Django as the web framework for the creative mind so I like to think of myself as creative, I'm a decent programmer, I'm good, I'll get on to me and what I do in a tick but that's the case I wanted to put forward.

Introductions, my name is Mark, I have been a developer since 2001 and like I said I'm an OK programmer but I have lots of different passions I'm interested in, I'm a musician, I write songs, I try and write comedy songs and also make pod casts. In my day job I work for Substract. I am technical director there; I'm in charge of the technical strategy making sure things work and keep running and stuff like that. We're do you sign led organisation and deal a lot with creative businesses so it's kind of the exact right job for me really.

And when I'm not doing that I run a thing called poddle which is a podcast and entertainment network that I created for me and my friends so that we can make stuff and share it and have fun doing it and start to build a community. I have been a fan of that medium podcasting and those various arts and pursuits since I left university in 2004 and I made my first pod cast in 2008 and I haven't shut my mouth since. I'm a fan.

So, previously on Djangocon, I was at Djangocon Europe 2 years ago in Poland and gave a 5 minute lightning talk so this is my second of any talks like this so if I ramble, if it goes off the rails, you have my apologies. But stick with me.

So, 2 years ago I talked about contributing to a community via open source and I was also about 3 and a half stone heavier at the time so that was - over a good thing, nothing dreadful happened - and since then I've been able to contribute to the community and be part of it so the kind of things Russell has been talking about I've been lucky enough to do, contributed open source code via the web-site hub and I've had that stuff come back, people have had improvements to make to my code and you start to build something really cool.

So, speaking of building something really cool because this talk was an open day I'm not going to go into big technical conceits here. But one of the things I wanted to do was - I make one of the shows I make is me doing other things and sort of talking while I'm doing other things, it used to be called thoroughly distracted, and I wanted to make that pod cast entirely on my phone so I wanted to be able to record it on my phone with a head set and up load it and have it available to people to listen to and I did it via drop box and I was able to build a solution thanks to Django in apart 5 hours on an evening, got home from work thought this is what I want to do and I was able to build a working solution where I could up load a piece of audio from a drop box folder and my thing would read the meta base using - some of the batteries included in Python using some of the extensions in the Python package or the {inaudible} I was able to read the meta data, get the audio, images and text and put it into the {inaudible} for the podcast. And one of the reasons I was doing that is because I'm using this amazing language, a language that not only has a lot of stuff in it that comes free but it also means because it is such a popular language there are APIs available for pretty much anything that you want to work with whether that's twitter and Facebook which I'll get on to in a bit or it's drop box or it's some Apple web service or something else, there are Python packages available to let you work with those things because you are using a popular language that makes a big thing of having an extensive library.

When you start getting into Django itself if you not familiar with Django you'll soon become familiar with the admin. It's a content management system that comes included in your project so when you start writing your model which is how you want to store your data, you start to get with only a little bit of work you start to get a full content management back end if you like that you can start to up load your content with and there are ways to customise that. That is huge if you want to be able to build something that is based on creativity because it means you are not having to build a content management system to do all of that stuff, you get it for free, that's one of the big bonuses you get over something like ruby on rails if you are making a choice for a web framework.

Django is really good at handling media not just on its own it does it really well but also there are some amazing libraries that handle that so we deal a lot with images and when you are dealing with images whether running a store or something else you might want to - for example with poddle we have big feature images for articles and with a library like sore thumbnail as an example that lets you deal so simply with images that you can say here's my main image I want it in lots of different sizes, I don't have to know what those sighs are beforehand I can tinker and try things in my image tags and see how it looks and make different response image size whatever it needs and you've got the tools to do that in an incredibly simple way with very little code no functional code, it's all in Django template tags and it's really simple so if you are dealing with images that is one of the great ways that you can get up and running really quickly so you can focus on making stuff rather than writing programmes.

So also you've got file browser which is a fantastic thing that plugs into your admin so if you want to re-use the same images or content, audio, video whatever, if you want to put part of it on a store or link to it on a blog post you don't have to up load same content multiple times so with Django file browser which is an extension on top of an extension but very easy to use this gives you not only a way to select a file from a server be that your own server or be it Amazon cloud storage it also gives you the ability to embed those images in a text box, you can embed those easily select the file, it also gives you for images gives you more sizes so you've still got flexibility there and it's a great way of being able to manage the content within your project.

So, one of the things that I mentioned about twitter and Facebook is trying to build a community so when you've built something cool and something that you want people to look at maybe you want to say, well, it's a small web app and I want people to be able to log in and check it out and share and that kind of thing and one of the little batteries you can plug into your project is Django social. It's a lovely, very simple way of being able to authenticate your users you are not worrying about building authentication system. Django has a solid system to use but if you don't want your user to create user name and password because sometimes people are funny about that or don't want to fill in from they simply want to get in authenticate from twitter or Facebook and say I'm happy for this site to use my profile and come back to your site then that is a fantastic tool to be able to do that and these are things you can use with writing very little code because the point is what I was able to do with my drop box example is I was able to connect lots of pieces together so I wasn't really creating anything new, I was simply stitching things together so it made my ability to make stuff a lot easier and a lot simpler.

What's wonderful about Django itself is it doesn't prescribe that you have to build things in certain ways. It gives you a bit of flexibility to say if you want to write your views which is the functional views that maps URL to something that you can see effectively, if you want to write that in an object oriented way you are familiar with or write it more functionally you've got options to do that, you've got options to write things in lots of different ways and structure your project in lots of different ways. There are project templates that exist that are created in Django that allow people to have that flexibility so you absolutely get to build it your way by - it's configuration over I don't know what the ... any way - thank you, configuration over convention. You get to host it your way and there are lots of ways you can host it, I used to use Apache and my SQL and there are people in here sucking their teeth going ... but there are lots of different ways you can do it and Django lets you do that and I then started to use a host called digital ocean which gave me a bit more speed and I thought OK I'm going to learn how to do this properly using services like engine X, which is another web server, but the idea is because Django is abstracted from that, you can choose the layers that connect these different pieces together so you are not having to think OK it must be on - it has to be on Linux and Apache and started building my site on windows because I was a windows guy before I started and someone encouraged me to use Django. Window was what I used. So because Python as Russell says is cross platform it doesn't care what operating system it's on I was able to build my first Django web-site and get them running on a server so you've got huge level of flexibility whether you use Amazon web services to host your site or small host digital ocean, you want to put it on a shared box with lots of other web-sites you've got huge flexibility to do that.

And then once you've got your site hosted you can scale it up. Django is by default really scalable. It knows how to handle a lot of things but there are also lots of great system that's plug into Django to help with things like cashing so when you view a web page and lots of people are viewing the same web-site the same home page if you like, Django isn't having to constantly go back to the database figure out how to display it, put it into the template, it goes I have just done this I'll get it direct from memory and show it to the user and that can make your web-site incredibly fast. You have huge levels of flexibility in terms of how you can configure that caching.

It's not just about caching, it's about other things in terms of scalability but caching is one of those things people will talk about.

And you've got fantastic support community in Django. There are people who are incredibly helpful. You'll meet people here who'll give you all sorts of advice; people tend to be very open. You've got places like stack overflow and the Django users Google group is useful. Also the IRC channel and there is a small plug, set up a slack community, called Django launch, it's very, very small at the moment but it does mean - I use slack every day so it's a room I can sit in and if people have questions then we're able to hopefully answer those and at least commiserate and go yet I've had the same problem don't know how to fix it - but at least you've got someone to talk.

So that's it really. Let's see how we did. Not too bad at all I'm quite happy with that. I've got a lot of details here where you can find the various things. All the stuff I've been talking about the pod casting stuff is on a web-site called poddle and there is a load of code stuff you can see of mine as well and you'll find me bumbling around the conference.

So I want to say thank you to Django guys Djangocon organisers for letting me speak. I am visually impaired so this kind of thing is a bit daunting but I couldn't be with a better community and I couldn't be using a better platform, Django is my favourite platform to use. In my day job I use PHP, Word Press, I use a few other things, mainly those things but Django is always a thing if I want to build something that I care about in terms of or something that's a hobby, I go straight back to Django and I always will because you get stuff up quickly and you can focus on the stuff you want to do rather than the nuts and bolts of the problem. If you are new to it, get involved, it is so much fun, so thank you very much and I thought I would leave a bit of time if anyone had any questions other than that thank you.

{Applause}.

NEW SPEAKER: Thank you Mark. Does anyone have any questions for Mark? My question is you put up a whole bunch of great tools that you say are helpful to what it is you do. For an absolute beginner who've been told you want to build a web-site use Django and more or less nothing else. In my experience it is sometimes hard to find these tools in these communities so what is the best advice -

MARK STEADMAN: For me it was the Django project web-site, it was starting with the building of a small web-site with the poll and how you work with the area so I am a digital programmer but I didn't know how to work with Python this was the first time I worked with Python so it took me all the way through even on windows, it didn't judge me for trying to build a web-site on windows, it said it's fine we've got this and took you through the whole process so when I speak to people and they say where's the best place to start one of the few frameworks I can say actually start from the official web-site because the documentation and starter help are really, really good.

NEW SPEAKER: Anyone else like to ask anything?

NEW SPEAKER: {Inaudible} when did you create - a couple of month ago?

MARK STEADMAN: Yeah it was only a couple of month ago so it is still very small but if you find me around I can give you the details but I think it is Django noughts dot slash dot com.

NEW SPEAKER: Is that a hand going up over there? anything else? OK thank you. {Applause}.

Before we set up the next speaker we're going to see if we can get the light system a bit better. So with a little bit of pride and immense of pleasure I'm introducing Rhiannon, Rhiannon is a first year student at the school of mathematics and she is going to be telling you about how she uses Python to get a ticket to some here really.

[adjusting laptop and projector] It is the frequency.

Let's again, Rhiannon.

RHIANNON TITCOMB: Hi everyone, thank you for the opportunity to be here. I am going the be talking to you about bézier curves today, I will tell you about who I am and I am here.

I am Rhiannon; I am a student at Cardiff university and studying math. I am first year, this year I had a module called computing for mathematics in which I was taught how the use Python and various other things. This was my first look at coding, never done anything before, but I really loved it.

So, I did a piece of course work in this module and I ended up winning a Djangocon ticket as a prize for doing well in the course work, so that is why I have got a ticket.

(APPLAUSE).

Thanks. So, when I found out I could talk here, I thought it is a really great opportunity and I thought to make it easier on myself I would talk about what that course work was about because clearly it was good.

So, I am talking about bézier curves today. What is it? Well they curves obviously and used a lot in the computer graphics and drawing and stuff on the computers. I will show you a couple of examples. So this is quite simple bézier curve and this is a slightly more complex looking bézier curve. So to describe it, it is a curve starting at one point and ending at another and it is curvature is defined by control points which are movable and you can have as many as you wanted, that is what shapes it.

So, why are they used? Well they are the most intuitive for a user to draw in these programmes, they are very, it is easy to see where you can move them.

Okay. Yes, so, they are used a lot because they are really intuitive and to manipulate. I will show you an example in a minute. They are also really simple and quick for the computer to draw on, take it and compute it.

So I will show you a quick example of drawing a bézier curve. So, so there is a tool here that draws bézier curves, it is a control point and you can bend it and those are more control points you can see there I am moving it around and stuff.

You can do whatever you want, that is a bézier curve there, if I click on this now. Those that you can see all up and down the control points and those are what shapes it.

Yes, so, how do they work? Well here is where it goes to the maths bit! (LAUGHTER). So, hopefully this will come up. Yes. No!

It is based on something called the Berstein... polynomial ... the T as the increases over time ... so, this, you can't see it but it draws out, sort of shows how the control points can be reduced to draw it.

(APPLAUSE).

Yes. So this image here sort of shows how the control points are used to help sketch out the curves you can see as it increases it goes increases goes to each point and stretches it out. Very simply I am trying not to confuse you.

But yes. So, moving on to what I did? I decided to try and approximate these bézier curves and using an algorithm. What this does is algorithm takes the control pointings and finds the midpoints and then finds the midpoints of the midpoints and keeps going until you reach some point. At this point it will draw the curve however I chose to use it an approximation and draw the lines between the mid points so, this is what it will look like, obviously it is very rough. Yes? I will get ton why that is in a second because it is basically the more a control points that you have in this curve, the better and approximation of the curve that this algorithm will give because it will have more midpoints to find and more line segments to draw. You will see a better curve.

So, I decided to try and create this algorithm and use it and so, I made a function here that I focused on the bézier curves, I made a function that take control points and this bit here is just a sort of ... what is control points are, like the X and Y coordinates. Then next it calculates the midpoints between those, so it takes the X and the Y's of two consecutive points add the both the X's and the Y's divide by two. Then it does it again for the midpoints found and then once more, down the one point.

After this I used (INAUDIBLE) to plot all of the points that I had and lines between each the first and the last of each generation, I am trying to call it. So it will do the first control point and then the last control point and then the first of the midpoints and the last and keep going like that.

So, that is what I started off doing.

I then realised it wasn't very, it didn't show the approximation for a cue bit one, I wanted to see if it worked so I decided to extend it for a bézier curve with 5 points and I will hopefully now it will work and I will draw a bézier curve with 5 control points and then put those coordinates in and show you what it does and hopefully, you will be able to see that it is an approximation of the curve. I will draw a cubit curve.

I will try and translate this using the -- I have to do coordinates to make it work. I will write this in down here, the first one is, minus 8...

Minus 8, 8.

Then the next.

So there are the control points, the first control point was minus 8, 8, then the second one is minus 6, minus 4. And then the next one is, I will call that the 2, 8. Then 2, 1.5.

Then 7, 3.

So, from this, hopefully, sometimes it takes a while to load. So bear with me.

Okay so. This is the approximation that my algorithm has given. So you can see it is a rough approximation, it has the right kind of shape and it looks like on a really rough basis. So, yeah it works, awesome.

So. I have already mentioned how you can get better approximations of the curves by using more line segments and stuff. However I notice that whenever I wanted to try and do a curve with more control points to get a better approximation, I couldn't because I would have to keep extending my curve, it was very badly written and inefficient because I couldn't do it as many points as I wanted.

So what I have been trying to do recently and it is not complete yet, I am in the middle of exams. I have been trying to write a function that using an end number of points and it will still work out. So this is the code for this. Input a list of points as X and Y coordinates that is before, so however many coordinate points you have on that, it takes the X part and the Y part and adds those to a list and then it will plot those X and Y points. Then after that. It will however many X and Y values there are, it will take the first X value and the consecutive X value in that list and then find out the midpoint then add those as coordinates to another list of midpoints which will then plot. So, I am going the show you that as well. Hopefully that works as well.

I will just use; I have got some points in there already. I think I have got 8 control points there, so I will just run that. Show you what it does.

Okay, so you can see that it has got in red are the control points and in blue are the midpoints and you can see that there aren't any lines or anything there, that is because I haven't got yet to the point where it keeps finding the midpoints, I haven't got it to take the midpoints of the midpoints, that is the next step. I am on the way there. That is good. Then, that is the end. I would have on the screen my e-mail address if you wanted to contact me and I link to I have got all the files on-line. So if you wanted to, you can down load those, I have put the piece of course work on there in case you wanted to read it. That is it, thank you very much. (APPLAUSE).

VINCE: A tremendous well done in dealing with a situation, you did exceptionally well. Any questions to ask for Rhiannon?

FROM THE FLOOR: I don't know much about the bézier curves but what is the motivation behind using ... as oppose to using splining?

RHIANNON TITCOMB: I don't know much about anything about splining. For me I wanted to use it from a mathematical point of view, which is interesting, as a maths student my work focuses on that. In terms of why it would be a good way of producing it? I don't really know. I think it is you can use it to draw, draw the curve ifs you implemented the (INAUDIBLE) I mentioned but I don't know if that is the most traditional way of doing it. It is a way as of a maths student I thought it would be a good way to take that. VINCE: Anymore questions? Let's thank Rhiannon again.

(APPLAUSE).

DANIELE PROCIDA: If you would like to find some nice fresh cold water for your bottles go back through the foyer where you came in where all the sponsors have set up and in the corridor just off there is the restaurant and at the back of the restaurant is the water cooler so we'll get some more signs out as soon as we find some tape.

I'm very pleased to welcome Tom Bakx who is a doctoral student here in the school of astronomy and physics. He is from the Netherlands studying here for the next few years and he is going to be talking to you about his work in Python and astronomy.

TOM BAKX: So, in 1995 a group of research astronomers used the Hubble Space Telescope to observe a small piece of sky over Christmas, they didn't know any galaxies to be in this piece of sky and not even stars and it's a very small region they looked at, so optimistic people were assuming there to be say 100 galaxies in a single viewing and normal people were thinking about a handful of galaxies but researchers were astonished when they found out that 6,000 galaxies were in this small piece of sky.

So, 6,000 galaxies is a lot to process and a lot of data to gather from a single image and back then there was a lot of galaxies but now there are samples of 3 million galaxies and recently has been launched a satellite going to observe one billion stars and observing that many different sources is a thing you really want to reserve to code as going through it by hand takes a lot of PhD students.

About this image it's called the Hubble B fuel image and some images are so far away that the universe was a lot smaller when the light was emitted back and ever since the light has been emitted the universe has expanded and while the universe expands the light travelling towards us has expanded as well, expanded from bluer colours towards more red colours and if we're able to find out how much it shifts towards the red we're able to find out what the distribution through cosmic time is of the different properties of the galaxies.

Now to do this researchers use Python and other programming languages, especially the open source ones as astronomy has a very big community and mostly the easy to use ones as we're pretty lazy. So, what did the researchers find when they plotted the activity of the galaxies through cosmic time?

They found that we live in a rather dull time. We're at 13.8 billion years since the big bang but 8 to 10 million years ago when the universe was 3 times smaller than it is now star formation and other very active processes peaked and these can be seen in these galaxies here but optical light does not give the full breadth of spectrum of star formation. Star formation is very important to understand why we see as many galaxies with as many stars around as to day and it is crucial for our understanding of the entire problems. However when a star forms a large clump of gas falls together to form a smaller clump of gas, gas heats up, a star forms in the centre but cannot radiate its heat away because there is dust and gas in the way and the gas and dust absorbs the light, becomes warm and starts radiating heat radiation and this heat radiation we can see in sub millimetre wavelengths but as you can see from the optical image and sub millimetre image there is no way of saying these are the same images. The optical shows a lot of detail and the sub millimetre shows 5 things you can call to be galaxies and they don't correlate to each other, the bright box in the centre of the optical doesn't correlate to the bright box in the sub millimetre and this is the thing {inaudible} statistical properties Python enables us to use allows us to correlate the optical and sub millimetre in a way that we can be sure we're looking at the same optical. The very brightly coloured spot in sub millimetre is so far away it is a 8 per cent of the age of the universe it is now, it is forming stars at the rate of a thousand solar masses per year where around us nowadays we only see galaxies forming one solar mass per year, so this a very, very active region in the universe.

Not only is it very far away, it is extremely bright, but it is being helped by something, and it is helped by a galaxy that is in the way, but instead of the galaxy in the way absorbing the light in the line of sight, the mass of the galaxy that is in the way bends the light around it and makes it appear less like this high resolution image can show us. This is recently been observed with a large observation array and Chile and it shows an Einstein ring as it is called of the lens image of the galaxy behind, image has been magnified 80 times and that allows us to peer even deeper into the universe. My PhD thesis is based around finding as many of these as possible because we can do some amazing science with this.

These galaxies are very, very heavy intense phase in their lives especially the far away ones and they're bumping into each other a lot. Like these galaxies these are 2 galaxies - in the midst of a merger. You can see the 2 spiral arms of the different galaxies swirling round each other and the structure they had before, 2 spirals has been destroyed you can only see the small rings next to them and it brings with it a burst of star formation. Now we understand star formation a bit and can trace it from elementary physics but if we want to understand how it really works we have to use Python in a different way to data analysis. We use it to simulate and this is a simulation of the Illustris model where we can see on the left hand side the optical light, the light we were able to see with our telescopes and on the right hand side we see the gas intensity where there is a lot of gas where a lot of gas is able to form. And as these central 3 sources, or central 3 galaxies move closer together they come entangled and the value you want to look at is rather small but says SSR as the third digit and that means star formation rate, covers around 100 the entire time but as the merger starts to occur you'll see it will peak to around 200 causing a phase of star formation throughout the galaxy, the galaxy we just saw in the sub millimetre is probably in the midst of a merger.

This is a small phase dwindling right now as we move closer to the present time, the number dwindles and falls below what it started as becoming not spiral galaxies but elliptical galaxies {inaudible}red, and dead galaxies because they don't form any {inaudible} and these correspond very well to galaxies we see today.

This is only a snapshot of the big simulation which I'd love to show you now because this simulation is of a cosmic scale, it shows a very significant piece of the universe and what you can see here is dark matter distribution within it you can see sort of {inaudible} light structure of the dark matter and the dark matter is {inaudible} like the web indeed and consists of 5 times more mass than the normal matter we can see. However it's invisible to optical light, doesn't interact with it, so the only way we're able to detect it is either by simulations or by the way it curves light.

Now {inaudible} the gas inside the galaxies and you can see there will occur explosions which happen at the centre. Supernovas exposures very big galaxies form very big stars; very big stars live a short time but go out with a bang creating heavier elements. Heavier elements we need to understand the formation of plants to understand the formation of water and perhaps the formation of life. In a nutshell, Python helps us understand the data we're looking at to make sure we get the most out of it and helps us figure out if the theory we're thinking about actually corresponds to the data we get from our measurements.

So, I think Python is a tool to put us in a bigger, in a cosmic perspective. Thank you. {Applause}.

VINCE: Any questions for Tom?

NEW SPEAKER: Wonder if you could dig into a bit more detail how you're using Python, it is numerical computation or how the graphics are rendering or -

TOM BAKX: Python is able to do all the things I showed you here but I assume I didn't use Python for the simulation I showed you - just for some cool images, probably used C {inaudible} because it's faster but you could use Python.

NEW SPEAKER: Specifically you say you use Python to crunch the data. Do you use any packages like I don't know {inaudible} or do you write everything from scratch?

TOM BAKX: There is a module called astro pi(?) and I use that mostly to {inaudible} data. Let me grab an image ... what happens here is if you were to just state the peak value of this image, you will get a less good estimation of the intensity the source has so you have to assume the profile you expect it to have then fit it and I use Python as a function to {inaudible} but I use it as a means of getting the shift from the red out of these galaxies just with functioning filling and do that and I use LM fit which is a different model but I'm not really into the module heavy stuff in Python. Thank you.

NEW SPEAKER: When you do simulations do you do everything on your local machine or use a computer and which set up you use like purely Python or you have some more supporting programmes?

TOM BAKX: The stuff I do myself I am able to do on my mac myself because it is not computationally heavy, it will take at most a couple of minutes, enough time to get a cup of coffee, but for example if you want to do large scale simulations you really have to go to super computers. It would take for the simulation I showed in the end to run it would take 2000 years on a normal computer but you are used to these {inaudible} because super computers? Yes, does that answer your question?

VINCE: Time for perhaps one last short question. Let's just thank Tom again. {Applause}.

A couple of announcements after we've finished that first session. There will be coffee in the foyer and otherwise one of the workshops is more or less started the Django 4 for PHP programmers. If you go to foyer where coffee is people will take you there. Finally thank all speakers from the session one last time. {Applause}.

(BREAK)

DANIELE. Okay just about ready to kick off. So, someone knows how to dim the lights so just to let you know, that lunch today is going to run between twelve thirty and two thirty at Aberdare hall it is on the website. I have mentioned it in the Djangocon twitter account as well. There is a map to get there. Go out of the back of the building, that side. Then you walk up the avenue until you get to the very end ... and then Aberdare Hall is in front of you. Check what times you are supposed to be in a tutorial or workshop, go at the appropriate time. The talks break will be as close as we can get it to between 1 and 2 I think is what is said in the programme? Is that correct?

Yes, the talks break will be between one and two, no talks between one and two, if you want to stay in the talks all day, that is a good time to go, if you want to go earlier or later that is helpful for the crowds.

If you have special dietary requirements if you have a vegan or if you have serious allergies, tell the staff there because some of the meals are being kept back for you, so nobody else takes your vegan lunch for example.

And that is all I have to say I think actually. So, we will David?

Yes, when you come in and introduce Cory while he sets up.

NEW SPEAKER: So he is going to tell us about a security nightmare discovered in the open-source product. Good.

(APPLAUSE).

CORY BENFIELD: I am using about eight devices. A quick warping I too too fast I have way too much material. I will have questions at the end.

You can find me at places, feel free to ask me questions at these places, if you can't grab me at the talk. I am kind of what you would call, an open-source software medium shop. I piggy back on other peoples work, I am a core maintainer on the requests project.

I am also on the ... three project and it is the most important project you have ever heard of. Maintain a few projects of my own that you don't care about, I maintain hyper, Python's only http2. I have done various bits of open-source, landed patches in open-source projects, some of them Python and some of them not.

I am going the tell you a story this is a story about one of my experiences with handing a critical vulnerability ... it is called requests. Little, it is kind of a big deal.

Anyway, like (INAUDIBLE) title it is catchy that is CVE 2015-2296 this will be a blockbuster with rights at end of the month.

2014, a long time ago, the world was a different place, we were in London, where it rained as much as here. The world was wonderful, the requests project, we had butterflies and bunnies cavorting; do bunnies cavort? Yes. Cool, I am worried they might frolic. They are bunnies, they hop around.

Everything was all right with the world, more importantly the requests project by this point still a fairly notable, had not had a single security vulnerability, we were feeling super-cocky, then we discovered that if you configured requests to website A, which we will call good site, that website redirect to another one Chaplaincy we will we all evil.inc., we will send your password to there as well as good.org.

So, we felt (INAUDIBLE).

So, yes, we leapt into action, a well drilled project and so we filed for two CVE numbers, CVE common vulnerabilities and exposures, a formalised number for keeping tracks.

So we filed two of them.

At the same time decided we should probably have a protest for these, this is a good thing to have. This is a list to look at to work out what to do. We had no idea what we were doing, we are idiots we don't deal with this stuff.

So, we kind of just made this stuff up. This is the crux of this story is, this was the error we made, we didn't talk to anyone about what we do. The first thing we did right, put up contact e-mail addresses, if you don't put up contact e-mail addresses, people will report vulnerabilities in the public bug tracker. Anyone can read them., this means the evil happens can read them and say hey, that might be a problem, I should do something with that, then exploit the users, get into trouble and then get fired. Put out a public e-mail address, let someone contact you, could be a personal, doesn't need to be anything fancy.

Along with it, if you are sufficiently technically savvy, put up a gpg keys., who knows, that is less than half the room if you don't know anything about GPG it is probably not the end of the world. If you can have them, do, that is my GPG fingerprint.

We can do this quite right like even for fairly technical project, one GPG key per developer. Rather than having a central.

Reporting a bug, either pick the one that will respond quickly or send one mail to every developer, not the ebbed of the world, it is not great, we do plan to having a project GPG key at some point in time. Those two steps fantastic, hard core, everyone can securely report bugs to us, nothing can go wrong.

All right so we fast forward, early 2015, doesn't count as now anymore, but it is no ... set the scene, the morning of Saturday 14th March, weather nothing like this. There are no wading birds here. I woke up around 7:30 a.m., I rolled over, grab my phone and read all the e-mails. Sitting at the top of the inbox, was this. So ...

Coffee has got nothing on that subjects like for waking you up in the morning. That is way, way worse, so I am panicked then I noticed the time stamp. I was already 8 hours behind, no one in the team got the this yet. I was the first person to see it 8 hours after reported. I freaked out a little bit. Started my week end a little differently to the way I wanted it. Trying to read the e-mail on a device that was good at reading e-mail.

So, reporter did something right. I want to credit Matthew for this, he produced a tarred of detail, it was probably two pages long and can take a thorough description of what the bug was, why it was a problem, and reproducible working code F. you run this, you will see the bug and in fact because it is request he went so far as to write a web server that would reproduce the problem. If you report a vulnerability, I reported one recently, found the bug via code read, I spent hours writing ... so this was great.

One caveat there, our bug was a security leakage vulnerability, so leak information to other people, if your vulnerability is code execution, don't run reproduction code that is an obvious way for someone to take over your machine, so I wouldn't do that if I were you. I mean feel free! Just don't hold my liable.

Matthew was right, this bug definitely did exist and like most relatively experienced programmers at this point, I didn't stop to think about it. I just immediately charged into trying to find the bug. So, quickly jumped into the Python debugger, within 5 minutes narrowed down the problem. I will have an interlude, I don't want to go into too much detail. I want to say what it was, if you don't care, there is an okay to put on the screen, octopus, talk to her while I explained the bug.

If you set, if a website set a http cookie, at the same time as it redistributes to the website, we will assume it is belongs to the website that redirected to. It is made worse, that some frameworks will set cookies on every request sent. We would happily send your session to wherever flask sends you, the specks say you shouldn't do that.

Right fine good, so I worked it out. Didn't tell me long to work out what the fix was, it was a simple one. There is a take away, we hear in the news, hot lead or log jam or insert vulnerability here, there are weird overflows that you can't have in Python, so easy to thing these vulnerabilities don't happen and they do. But they happen because of simple logical errors this bug here would not have been caught by any type checker that is available in Python, both objects of the same type and passed into the function, we were passing the wrong one of two.

There is nothing you can do about this, code review, this function is a hundred lines long, didn't get effectively coded. We would have been extremely lucky. That is mostly where you get the errors from.

I found the bug, prepare a fix, ship it in the next release, great. The answer is no, the second you push a patch for a vulnerability you should assume that it is public knowledge, pain in the ass to look for vulnerabilities but quick to check a log.

The news is contained I know about it, Matthew knows about Is, Ian probably knows about it. He is in America.

But, for us to find out, order our code base. The second I push this they know and can start exploiting the users. So remain calm and level headed.

I am not good at calm and level headed at the best of times, this is not one of the times. Up until this point I did everything right, I could have pat t myself on the bag, given a most mortem. I said don't do any of this, I am terrible.

Don't let panic overcome you and don't move really fast. We prepared a patch, pushed it, cut the zero relux, pushed it to PIPI and contacted ... there is a stream on the, you will want to update your stuff.

That kind of feels like a success because we are like yeah, went from a report to a fix in twelve hours, eight of which I was asleep for, soy feel pretty good about myself. This was so quick we didn't have a CVE number for this. We said security vulnerability, CVE pending. I did it the next day.

So here is the problem, that is wrong, we did a number of things wrong we should have done better. So, thing one, released on a Sunday. Don't release on weekends it is a terrible idea. Firstly, lots of your users will be businesses, most businesses don't work weekends and they are not paying attention. Attackers do not work nine until five -- they live in basements with plenty of hours and ... releases patches on weekends gives attackers a day or two of head start on your vulnerable users who are businesses who don't know what they are doing., businesses are (INAUDIBLE) and those require that pretty much the second someone becomes aware of vulnerable they have to assess who is at risk, I have people following us on twitter. Forcing people to work on a Sunday doesn't enforce ... release on weekdays, preferably Tuesdays.

Don't release before you have a CVE number, this gets into the same concerns of businesses and complaints, CVE's are how businesses track vulnerabilities, whether they are exposed to them. If you don't know how to get them, Google "how to get a CVE number and great."

I did that, it is fine, super-easy to get, not that hard, make sure you get one.

If you have downstream redistributors, warn them ahead of time. This admittedly might have been a privilege for larger projects, you might have surprised how much have got them ... if you have them, find them out, form a relationship with them and make sure you can warn them ahead of time. If possible, and you are GPG savvy, have a conference, so you can definitely contact them and only them. That way you can coordinate a release over all the channels, minimise the risk to the users ... they are people too.

Also, if you put ... (NAME - INAUDIBLE) users to hate you, ... they write nasty e-mails.

Right, we will speed through. Clear what versions of software are affected and how to fix them. Just upgrade. We had to request all the back verses so people know, most importantly have a public policy about the stuff, have a document, here is who you contact, how you do it. Request, promises to get back to you within twenty four hours we will acknowledge the e-mail, then a separate time how to push for it. If you don't know how to write one, steal ours, look for the vulnerability section, steal it, maybe change the keys in the GPG keys.

Number one tip show, scar, this is good, be prepared for this sort of thing, it can happen to you, you want to know, hey, I saw a talk and maybe there was some good advice in there. Cool, I am done. Thank you very much. (APPLAUSE).

NEW SPEAKER: Thank you, sorry to cut you short.

Whilst Amit comes to set up, many a couple of questions?

Anybody?

FROM THE FLOOR: I have got one.

RUSSELL KEITH-MAGEE: Fantastic, thanks very much for that, by the way Django has one of the policies in Wales, I would be interested. Any comment how it changes in?

CORY BENFIELD: It was, vulnerability is not disclosed to you, you have become aware of it, already exploited. Everything moves much faster, my ... it was entirely inappropriate in the case of a well disclosed vulnerability, again, Alex has a really good log post on this, I recommend you read, I steal most of my good ideas from this. This is one of those, get ready to check it out.

There was another question?

FROM THE FLOOR: So first off thank you for riding requests or. Have you had a chance to put your new improved process into action?

CORY BENFIELD: We have not, I think the best kind of work is work that I do, that I don't have to use, I am happy to waste the 8 hours.

NEW SPEAKER: Any other questions for Cory?

Thank you very much. (APPLAUSE).

Amit is going the set up for the next talk, while he is setting up. I would like to mention two people who are doing a really great job for us here today, Sheryll and Hilary, here at the front from Action on Hearing Loss who are providing the speech to text transcription, so thank you very much.

(APPLAUSE).

Is this more difficult than most jobs?

Yes, it is.

So, they are doing a really valiant job and it is fantastic, if you know, if English is not your first language, if it benefits you to see this, make your way over here so you can see the screen, I hope that it is proving useful to people.

If it is proving useful to you, mention it next time you go to a conference, there are people with hearing impairments who don't go to conferences because they feel they don't hear things. So, thank you very much once again, they will be here with us for the next 4 days, I hope we don't wear you out completely. So thanks. (APPLAUSE.).

Very pleased now to introduce Amit Nabarro who is going to talk about RESTful APIs with Django. {Applause}.

AMIT NABARRO: Working? No? Is this working?

NEW SPEAKER: For the video...

AMIT NABARRO: Hi everyone my name is Amit. I am going to talk about RESTful API with Django today. A few words about me. I have been doing software development for a living for far longer than I care to admit. 7 years ago I read a book about ruby on rails and a saw a comparison with Django and I pretty much never looked back. I only work with Django these days.

So, anyone who has done anything with Django in the past knows that traditional Django what I like to call classical Django is based on a concept which is called model view control or a variation of it. Most of the frameworks out there today use this model and Django is no different. And if you have written even the simplest example in Django you know that views return with email, you send a request and they return HTML - - but that's not always enough because this was maybe could 10 years ago when only browsers taught your application but today other kind of devices talk your application, mobile devices talk your application, machines talk to your application applications, other servers talk your applications and they don't always want your fancy beautiful HTML you created, they just want raw data because they have their own presentation data their own thing they do with your data. So the traditional or classic model which Django was built on isn't that useful in those cases.

Well, before I go further I'd like to remind you something which you all should know probably very well that the internet works in a very simple concept which is called request response. You fire up your browser, type in a URL, that request goes to the server then the server generates a response, a response goes back to you. The protocol this whole thing works on is http. Http is a text protocol, it happens to use HTML in classical Django - - - but it doesn't have to use HTML, it could use other text based formats as well. It could be used Json, XLL, the cool formats you came up with.

So since you can do that with http and since you can return other forms of data rather than just HTML, you could be using the same mechanism in Django to return your data.

Now, why would you want to do it?

The first thing it allows multiple platforms to access your application. For example mobile devices have their own user interface, their own platforms, they're not interested in your HTML, they just want to get raw data then display it for their users, so they would like to request data in the format of Json most likely.

Other machines, servers that may access your own application and down load information from your database. That's another good reason to do it.

Modularity, separation, we're going to talk about this in a little bit.

Then the last part which is also very important, it makes it very easy to develop single page applications. Anyone who doesn't know what those are, those are Java script applications, called skateful application, the entire application down loads to the browser in one request then all subsequent requests are on HS. You've all seen those, they're very common these days and they're very useful.

So, in order to have your general project expose these capabilities you need to develop something called a web API. What is a web API? I ask that question on the web and I found this definition: an application programmatic interface to a defined request response message system.

That's great, that fits really well with the basis on which Django works on the http request response message system. So all we have to do now is to modify our views instead of returning HTML get them to return something else, return Json for example.

A lot of you might say OK fine, I can do that no problem, I don't need the risk framework or a third party tool, I can return Json no problem I can modify my review and return Json and here is a good example the most simple example which are a function of a view, returns the first name and last name of the user. This will work, this is actually valid code, and if you make a request with any tool to your web server you will get a Json which replies to the first name and the last name of the user currently in session.

Now while this is a very valid example it's really not very useful because most projects are much more sophisticated, they require a lot more capabilities than just something simple like this and for that reason most projects that involve a web API end up using some kind of a web API framework.

Now I've mentioned the term REST and RESTful API, ever since I started talking. What is REST? REST stands for representational state transfer. It's a software architecture style consisting of guidelines and best practices. REST is not a technology. Rest is an agreement. It's a contract between a client and a server. I will serve you a restful API. OK I know what that is, I'll write the client to consume your restful API.

And there are multiple ways to develop web APIs today but REST is considered one of the good ones. And in general alone there are multiple web frameworks that support the model.

Here is a couple of examples and principles of REST. The most important principle of REST is every resource is unique and in REST terminology every request you make is made to a resource, a resource can return data, and here are some examples of URL that is a unique URL for server, requesting first user or requesting user based on first name or requesting - here is an example, the last one is an example of a nes {inaudible} resource it's a resource that belongs to another resource - this is outside the focus of this talk.

Another important principle of REST is that the inter actions are state less. There is no state carried from one to another. Nothing is saved on a server. Everything has to come either as a body request or in the URL.

So, I said earlier that yes you could write your own views that return Json or return another form of text based response but if you are going to develop something that is a little more sophisticated and simple user name than the simple name of a user then you will want to use a web framework which a - a REST framework which will give you a lot - now what can you expect from a REST framework?

The first thing you can expect is see realisation, take your data, see realising it, Json. HTML very important. Pagination, you have a lot of data and you need to paginate through your data you can't obviously make a single request from all the data in your database because that will take for ever. Validation another important thing especially when you submit data if you are posting or putting data and want to validate it that's another thing you can expect from a REST framework authentication anything to do with users logging into your system. Authorisation anything that has to do with what the user is allowed to do on the system. Throttling, throttling deals with managing too many request's when your server is bombed with requests you want to slow down, you want to allow users to have only certain amount of requests per se second or certain amount per millisecond that is where throttling comes in. Caching very important. If you have a single request data being called multiple times you want to cache your response so you don't have to get it from the database then serialise it again because it's a very good concept and it's a very rooted in Django itself. API discovery that's also very important. If you are developing an API which goes to another third party user which will use your API you want that API to be documented and discoverable and you can expect your REST framework to expose its own API or its own scheme and say OK these are APIs I'm exposing, unit testing if you don't write unit test you should be ashamed of yourself. And a lot more ...

Existing REST frameworks are for Django. The first is the Django REST framework DRF, written by Tom Christie, excellent piece of software, fantastic, huge user community, very well documented, highly recommended.

Django tasty pie just as good just as well documented, huge user community, actually it's even been around longer than DRF, and it's a very good option. Django piston is a third one. I wouldn't use it. Django piston is dead; it hasn't been maintained for a while, documentation code, just don't use it, if you happen to run into it move on.

Here's an example of something which takes a model, a Django model and turns it into a resource. So I mention before that in the REST terminology we consider our data as resources. Here is an example of taking a book model or a database model which represents a book and turning it into a - this is all you need to do in order to get, take a model and turn it into 4 lines of code. Obviously a very simple example, probably more simple than the tutorial you're going to see in tasty pie, this is done in tasty pie not DRF, but DRF is very, very similar. And what will happen is these 4 lines of code is going to turn your model into a resource which you can perform crude operations on, create up-date - all that in 4 lines of code. It's pretty cool.

Working with SQL that's not a problem at all. REST is not limited to relational databases. You can wrap your data sources with a REST framework to any data source you are using whether it's a relation database like {inaudible} or using mongo DB or any other thing. In fact on the fourth day of this conference, there is a workshop that I'm doing on mongo DB and Django and I'll be touching that subject in very detailed way.

That's it. Pretty much. {Applause} I would like to say one more thing. REST is a very big concept and I can probably talk about it all day but given that it was only 20 minutes I tried to condense it as much as possible and give you an introduction to it but if you are interested in talking more about REST I am going to be here for the REST of the conference.

NEW SPEAKER: Thank you very much. Questions if anyone is interested.

NEW SPEAKER: Which of the 2 recommended frameworks do you think is most likely to end up in the core of Django?

AMIT NABARRO: I don't know if I can answer this question unopinionated. You want an opinion I have no idea. I use Django testify {inaudible} for familiarity and no other reason. Django framework is fantastic but it's my personal preference.

NEW SPEAKER: What's the main reason you decided to use Django and not rails for your APIs except to {inaudible}.

AMIT NABARRO: Before I was doing computation I was using Python so it was just easier for me to do it but if you look at - I like to look at trends and Django is really trending up and ruby on rails is ... that's not my opinion ... {applause}.

NEW SPEAKER: My question is - ruby on rails doesn't have much support on-line {inaudible} engineering things like that, whereas Python has a lot of support on things like data management and engineering things like that. What's your take on that like?

AMIT NABARRO: It's a known fact Python has way more third party modules than ruby so you are more likely to find something in Python than - I can give an example, just not too long ago I was writing a driver for an engine and it was talking something called mud bus which is just a protocol for motors to talk and it's based on http and {inaudible} wait a minute...motor bus ... that was it. Anything else? OK thank you very much. {Applause}.

DANIELE: So while Yamila sets up, to let you know that there is a jobs fayre with the sponsors in the foyer, so if you are interested in talking to them, now might be a good time. Otherwise they are going the be here all day, you can talk to them at any time.

If you didn't get the message earlier, whatever it says on any piece of paper you may see, lunch will start at 12:30 that is the earliest we are doing lunch. If you are only here for the Open Day, sorry, we are not able to provide you lunch because we have only provided lunch for the people who signed up for it because, those are the numbers we have, but there are plenty of places to grab a bite to eat nearby.

NEW SPEAKER: Okay, the next talk is by Yamila, Ten lessons learned during the a big Django project. (APPLAUSE).

YAMILA MORENO: Thank you for coming to this talk. As I said, it is about ten lessons I learned in the Django project. My name is Yamila, this is my twitter and any other resources I work Kaleidos. So this talk about the good and the bad decisions we make in our big project built in Django, it was a complete renovation of our website, roadmaps, meant for all of you who have facing a big project for the first time or maybe starting with web development if you have working with Django for a while now, you h identify issues and the questions and other things.

So it was a traditional brand with roadmaps relating to our massive oil company in Spain and we had to, to repeal the brand and it was traditional brand we needed to keep the essence, so it was quite difficult. We decided to focus on five main features were to tourist, gastronomic cards, search engine, power map, planner, editorial content. So how big was it? We had to handle eighteen thousand cards, this among is data is not big if you any of the big data stuff, we had to show all the cards in a map at the time. It was hard. Firstly, took about 18 months then took time transferring., so today, continues to evolve.

So, the original team was formed by two user experience one designer, one front end, six back end. I was part of the back end team. As well, managers, relation team, the marketing, so, lots of we had to take on board lots of different opinion to do the platform and as we were, we had this system, two weeks, so we had forty sprints, six hundred US's and thirteen hundred points.

With all of this, this was a project. One of the first things that you have to think about whether the project is big or is small is about the architecture, when you have a small project you can keep the proposed architecture of Django, with the back and the front coupled but a bigger project as the former speaker talk, will face a new challenges and bigger challenges and which will require a stronger foundations. For instance, in (NAME - INAUDIBLE) we are starting to develop the project and six months later, we were asked to make, mobile version applications, mobile applications, export the data to other applications and we were unprepared we needed to make a lot of changes to go the new requirements so the lesson one would be completely separate API and front projects. The API is stable and the front projects are changing every day.

Apart from the architecture, there are also places you have to be ready, so you need to make it ready for any new requirements for instance, we had to integrate the Google API, Google maps API so took this and isolated in our different group and two months later, we changing all the maps API because we can. So we just needed to change in this module. So, we thought of course that the effort was worth it. But, with great power it comes great responsibility. The everything, factor in everything, in the way of the dead lines of the project or in the weight of being paid working with the team. So the lesson two would be ready for changes but don't over-engineer. It is difficult and useful for these to use some rules and, my rules refactor and abstract the second time you have the chance. If you are working, you have to, within the method you are working on, but the next time, refactor the method. This allows you to refactor.

So quick survey, raise your hands all of you who know about the loss of thermodynamics so, laws of thermodynamics, now rise your hands if you know the fifth law of thermodynamic yes, I was expecting that, it is common to forget these very important law which says sooner rather than later you will make a bug! it is not me, it is the dynamics.

Being that true, the lesson is quite easy, test, test, test. The key is that if in a big project many people are committing to the call. So you need to be sure that your changes won't affect the behaviour of the application. So you need test. You need to know that your future changes won't affect the previous behaviour, so you need to test.

In, we wrote tests but not many. Not disaster, but with some of the changes something or adding new features because there are dragons, we didn't trust ourself, which is normal. So says you have in Python you have a specialised framework you have to know about knocking, stabbing, spies you need lots of resources to make your application. Testing, you need to test the application.

One of the most powerful resources when you are working with Django is the community. In the community you have lots of libraries, big and small. You have I think I counted more than a thousands. So, some of them are probably famous Django framework, request, CMS we use all of them. In as well as in smaller libraries. All the resources help you achieve one important goal which is not reinventing the wheel. Sometimes the library is too big and, very little part or sometimes the library doesn't do exactly what you need or maybe you need the absolute control.

So in those cases, you can take advantage of the adjustment of the flexibility of the power of the Django, and Python. You just need to know the internals and do it yourself whenever it is needed. It is difficult to know when to use the resource or when to do it yourself. In the map, we need everything with the map we didn't found any library who could deal with the front with eight thousand cards which were points of interest. So we needed to class them in the back end and then show in the map the, for instance, the literal content was made with Django CMS as it is open-source, we used at the base and made some widgets and we changed some functionality, there is no one answer but just to try to be aware of the context and change the approach.

Besides the, there is about, the project when you are working in a big team, for me, it is 5 people is a big team because everyone is changing. You should follow some good practices for instance, this, use a distributed source control manager. If you have been working with, you know this, but if you are starting you don't know you really need this., (NAME - INAUDIBLE) so ...

But, whatever you are doing, you need to use it. For us the lesson was not to use the source control but to add the widget to our work flow, so we started to use requests as the basic request to have procedure, so we created a brand, then develop anything there and then when we felt comfortable with the code, the test and the recommendation we made a request. This request would be reviewed by another colleague who would so, this colleague needed to understand, to test the code and make a widget or maybe accept a request, at least two people reviewing the code before adding to master brands which means like going the production.

So use pull requests as the basic procedure to add features to master. We thought it would save time in overheads. We, we were making better code for the very beginning thanks to this rule.

Very often overlooked issue is documentation. But in a big project you, even in a small project but a big project needs to be documented and well documented. But the lesson, so, you need, in my opinion in my experience it is important to have two different interpretation, one is meant for the developer who is are going the deal with the deployment and the API for those consuming the data and they are so worried about requests and responses, and something like that.

But the lesson is not only documentation which is important but documentation driven development. The first thing we do, to write the documentation to agree with the front team which are the first steps consuming our API and then once agreed, we could do the implementation, similar to the sign by contract but focused on a higher level and it worked very good for us.

Then finally the ten lesson, when you are working with Django, you are using and you are clever because you know about thermodynamics you are using the open-source resources. These resources make your code better, make your life happier. So, it is important to give back something to the system to keep it alive to keep it growing. You have lots of many ways to make as much open-source as you can. You can report a bug. You know how to report abnormality issue now. You can fix a bug, you can write a tutorial, you can just help someone, you can release a library and for instance in Kaleidos where we work, all the customer haves to agree that we are going to release code. From the development, from the project. Also in Capacity we have our own project and we are very gad and proud to share with the community. If we want to share it and not be ashamed of our code. More important we want the community to work with us, we want to make it easy, for that we need to follow those rules and many, many other rules. So anyone can run a test and they know they are doing things okay, they can read the recommendation and they know and they can understand everything, so, so it would be easy to make the system grow.

[baby crying], he doesn't like to ...

Yes., beside the ... you can think about many, many more rules, maybe you think all of these apply to a project and they do, think about Django and the community and the research and the recommendation, you can maybe approach to make the platform or the project or whatever much better.

I think that is it. Any questions?

(APPLAUSE).

FROM THE FLOOR: How do you decide that a piece of software can go open-source for a project?

YAMILA MORENO: From my company point of view, two ways, the first is if I am using Django's framework I can then use my working time to improve the Django's framework on so this will be the first thing. It is in the my spare time, my project is taking advantage, so I can put my effort my working effort and the second one is we prepare the code to have libraries. So I can make an agnostic library, my project can use the library but not release it. If you can't release on the project it is, the most important sorry, the most big, the biggest way to share.

RUSSELL KEITH-MAGEE: Actually following up on that, can you, have you got any experience with, any advice or experience with convincing clients that it is okay to work with the code., it isn't hard to convince someone to say it is a free library, when we build your project we are going the open-source a large part of it. A lot of companies get nervous you are going to give away secrets.

YAMILA MORENO: The contract for us is mandatory, you are working with us, if you don't agree to this, this point because we are an open-source company, so, it is a basis. We, the code that we make belongs to the customer, so, the critical points of the business logic belongs to the customer, but we are making, need a special authentication and we make a special library for this or we make some extensive library that is not about the logic. We release it and they agree, they are very happy with this because usually when you are going the set a contract with us, they know us, they know what we do and they really like to be part of it.

Maybe coolness for them.

NEW SPEAKER: Was going to ask, so when you have got deadlines looming and you have to work faster and faster, and fleshing things out, what gets cut of the best practices, documentation, practice? What can go?

YAMILA MORENO: Thank you for that question! (LAUGHTER).

NEW SPEAKER: I am the awkward client.

YAMILA MORENO: We call it the hour of the brave where we stop doing tests. Thank you!

No really, usually we just come to agreement with the customer, okay we will have this, this next release for you but the next two weeks are for us. We have, we are writing rubbish and we need to rewrite it. So that really is, we will fill the, we will accomplish the dead line but then you owe us two weeks or whatever to rewrite properly. So, it depends and documentation it is for us, for me at least is the most difficult thing to do because I don't like it!

NEW SPEAKER: So the technical dep, but you need to -- find any issues, explain that to clients sometimes?

YAMILA MORENO: Yes. Not only that because we also have developed a little project also in Django, which we call ... (NAME - INAUDIBLE) whatever we increase our technical depth we put a monster in our, in the dungeon, so the client can see how it is growing this. So it is a nasty thing, if you want to clean this dungeon, you need to give us some time but the customer also has a word in the decision.

NEW SPEAKER: How do you deal with situations where you will only need to use some of an external API for example, so you build a small library that could be open-sourced but it doesn't do all of the things that other people might want to do. In my experience a lot of developers are nervous to open-source that handle 10% of the API, they think they will get complaints from the community it is not good enough.

YAMILA MORENO: I know, I mean you can go into ... maybe the 80% libraries that you need for once or twice and hopefully someone can take advantage of this and whenever it is possible we try to make it agnostic and useful but it is not always possible.

NEW SPEAKER: You would say release it rather than sit on it?

YAMILA MORENO: You don't need to take with us, if by chances, from anyone, take it.

NEW SPEAKER: How do you make sure that your internal documentation stays up to date with the code you are writing?

YAMILA MORENO: What happened with you? This is my first talk ever ... (LAUGHTER).

There are some ways, you can know, I can be sure but for instance, we also do refactor documentation, we take the time to review the documentation, for us, a very important resource is to add a new member of the team so, take the documentation and see how far can you get? So, when he is, from he or she is a start okay, my documentation starts because always it is the documentation, it is not the developer. Most, 90% of the time is that I wrote something not clear enough or something like that. But, you have to refactor also your recommendation.

DANIELE: Thank you.

So, just a reminder that lunch will be available until 2.30. Stay for the next talk and then we'll take a break for lunch from here. Unless you have an appointment booked with a workshop soon after lunch in which case you might need to go now. Lunch available until 2.30.

NEW SPEAKER: Hello everybody. So, our final talk on this section, please don't go away for lunch, there will be plenty of food left. So we're going to have one more talk by Raphael on cyber ponies. Thank you very much. {Applause}.

RAPHAEL BARROIS: I have one option one is I go very, very fast and everybody will be ready soon or I can try to present things in an able manner which I'll do. Today we're going to talk about {inaudible} cyberponies, Django and machines. What do I mean by machines? Where I work we are entering car sharing system in Paris, Lyon, Bordeaux, soon in {inaudible} ... some are cars so they're good with some systems and units and some are charging points so you get a big {inaudible} ... so devices I'm thinking of, you can think of all devices, you can think of traffic lights or small {inaudible} to know what temperature is so what I'm going to talk about days vices that are full of sensors around them that you can promote {inaudible} in terms of traffic light green ... and local interaction like swiping your card on a point. And all those 3,000 cars and points with Django and I'm going to explain {inaudible}.

What do we need to do? First you need to manage {inaudible} - your inventory, what devices I have {inaudible} when did I buy them {inaudible} new devices as they come out of the factory or when they come in. Hopefully you will have to push from {inaudible} new features, basically because we want to {inaudible}.

Monitor devices. Sensors, things coming up, record everything that happens, everything, road, or ...

And you want to detect issues, sometimes you'll lose a device, drops off or perhaps the {inaudible} is broken or you want to know about it and you want to be able to tell the officer oops there is a problem you cannot transfer this car it's not available right now.

Actually use the device, {inaudible} on your board, it is there, shiny and you want to interact with it.

So you might want to aggregate information from all your devices and put them to your end-users and you want to be able to convert user request from your web-site from your API, from your internal application to actual {inaudible} the devices.

Inventory ... OK the device will {inaudible} inventory,, once a year when replace port ... but you've got serial number, model, manufacturer, mac addresses, registration number, lots and lots of numbers, versions and information to keep, and it has components and subcomponents, when you have a car, hundreds of items in the car and you want to track everything to know perhaps if one batch is malfunctioning.

So just a model, kiosk, that's my internal reference and it's maybe a new kiosk model ... {inaudible} small kiosks, {inaudible} then you've got your actual {inaudible} you install somewhere ..., it has a model, belongs to a customer, it's somewhere in France. Basically you want to describe your objects Django will be pretty efficient for describing whereabouts. Put them in and you already have your whole database. Just with Django.

So if you want to {inaudible} perfect. If you want history you can use various libraries for instance I think Django reversion keep track of changes, you can use libraries for recording building nice graphs. You can build a simple APIs {inaudible} information system can access that data. Your devices can push inventory database and request for API.

OK a few things Django cannot solve. How do you registering inventory. If the board is about to scan all parts all serial numbers and push - {inaudible} provided you have device and a huge {inaudible} ... have to scan them manually, and when you want to push on upgrade you want to push up-grade with Django, perhaps you say with Django I want the device to be updated to that from a version but actual {inaudible} on to the device is going to be something else so yeah you have to do something else.

I'm looking for some very, very important thing when you have lots of devices ... internet... {inaudible}. Security, you want something that says when I am here - it is the right device but it is giving you serial number, someone trying to fake it and when you send data to your devices you want to make sure it's the right data and it's not been altered and it's basically your proper data and someone is not {inaudible} ... that's a few important things when you will have to rely on other tools via Django for instance useful for your place, lots of tools are coming in recently. So that's more what it is what Django is, the perfect tool because it's {inaudible} devices, you don't want to run Django on your device.

You want also to monitor devices. Let's say for instance device crashed OK it crashes so {inaudible} it's not working. OK. What happened? Why? I need to get the last 5 minutes usage or things like that. I just push a new {inaudible} to all my cards, are they performing better, worse? I have to know that. Actually the devices are same source with the charge points, how do we know whether a car is parked in front of charge point? We need to get that information to know whether we can {inaudible} the charge point or not. We don't have eyes on the thing. Our sensors our monitoring - only way we can know what is happening in the interface. So yeah be prepared to get a huge amount of monitoring traffic. Sometimes if you get a message once every 5 minutes but 3,000 devices it means you get 10 updates per se second. Say you wanted to have one once every 5 seconds want to switch to {inaudible} some sort of system so take that into account when deciding your network and system. But actually if you just want to spread the load of {inaudible} images, send them to the database no problem, {inaudible} application servers running Django because Django is designed you may put several {inaudible} database so if your database is big enough you can {inaudible} ...

A table {inaudible} around {inaudible} lines right now, we can query it pretty fast, postgres is amazing {inaudible} ... elastic, choices to choose from.

That's what I advise is to get all your requests through Django that may if you want to get for instance information about how do the latest engine perform you can look through {inaudible} recorded, {inaudible} then ... monitoring through {inaudible} through Django ... put that in your report.

A few important things to keep in mind when you are distributing system. You will lose messages. At some point they will be full or have to miss messages or some will be broken, you will lose messages. Some messages will arrive late because while the car was parked in another {inaudible} then it has to send 5 hours of data at once. Sometimes you'll have some issues because your devices have clocks of their own which aren't synchronised so one says this happened 2 p.m. oh actually it was 2.30 when it happened. You won't be able to do anything.

So tips. When send message to database just put {inaudible} that may if you have lost an event well, you'll catch up later on next message.

Send the time since last change for all binary sensors. I know what happened when it happened and I can fix my data. And what we found very useful is to keep a last known state cache in the database which means we have one line instead of in? Device so it's much more manageable and have access to all the current state very, very easy. Good, reports ...

OK now we've got our inventory and we want to use our devices at this point. Yeah one problem you have is your users they want information about for instance {inaudible} home. Sources in their home. So data to provide a global view of the situation and they have lots of devices to consult that are on. Mobile phones, tablets, they've got their web-site, so you want to put all that information in an agreement so you use {inaudible} proper {inaudible} for err users where Django helps and you want to send commands to your devices when someone acts on its interface, participant, or when your computer needs to reboot or reset some device. Well, here it helps with Django because for instance let's say I want to reserve a charge point I want to make sure it has {inaudible} received monitoring message recently and it's not used but can create reservation but keep in my Django database for anything else then I send a manual command, blue, that's the logic to charge point and here use the users, I want a reservation, you do that from Django, check the form and you just code your process, {inaudible} user swiping a card.

Huge changes to solve. You cannot send request from {inaudible} to devices ... here it is more complex - split brain effect which is what happened last time it received message from device. Partial view world has changed. You cannot know what has happened you have to guess and have to design and keep that in mind when designing your apps.

That's all so do you have any questions? {Applause}.

RUSSELL KEITH-MAGEE: Are you running Django on both ends of this on the server, devices or just on the server...

RAPHAEL BARROIS: Just on the server. {Inaudible} on the devices but Django {inaudible}.

NEW SPEAKER: How do you send out {inaudible} devices the software permit of it.

RAPHAEL BARROIS: For now we're using the {inaudible} packages so {inaudible} should be now using version of that much of our meta package so it starts a new package with all occurrences of all versions. It's not perfect but it works for? ? And we're looking at oceans like {inaudible} snappy which is going to have {inaudible} devices.

NEW SPEAKER: How you deal with tests like for instance you have a device that sends its state, how do you make sure if you use {inaudible} for instance that the device doesn't change over time that you're using the wrong {inaudible} for instance?

RAPHAEL BARROIS: We are building tests where we are running basically the whole ward on the {inaudible} we basically send fake messages to fake device {inaudible} sends a message to the app and back again and so we can {inaudible} like that and build for complex scenario but it's kind of tricky and so we've had project to design simple ways of running full integration tests. We don't have to run the actual device code which will send {inaudible} want them check the device runs properly and we use some database working to ensure we don't have {inaudible} for the same thing at the same time.

NEW SPEAKER: How do you detect that some sensors are broken?

RAPHAEL BARROIS: Broken sensors for instance we detect that a sensor is changing states too fast, for instance when a charge point says hey people have been connecting disconnecting 1,000 times in one day you think it's not physically possible so it's broken. {Inaudible} at all in the time where it should have because we have a few different {inaudible} for instance got to open the charge point to get access to the cable so never opens while you connect disconnect the cable probably is it's broken {inaudible}.

NEW SPEAKER: {Inaudible}.

RAPHAEL BARROIS: Well, for our next generation vices using web {inaudible} which allows us to send messages direct {inaudible} registration time for comments we want to send to devices. {Inaudible} more connected state. It can work for charge points for the cars. It's going to be slightly harder. Good. Contact information and if you have any questions about this or perhaps {inaudible} I'm making and perhaps {inaudible} feel free to ring me any time. {Applause}.

NEW SPEAKER: We're going to break for lunch now. Be back with the timetable by the time you come back.

(APPLAUSE)

(LUNCH)

DANIELE PROCIDA: We are going to give folks some time. I am Adrienne, I will be speaking, thanks for coming to my talk.

Starting in about 5 minutes.

All our technical crew in place, ready to go, you are already to go with the projector, smashing. Welcome back, I hope you had a nice lunch. If you didn't have your lunch you have got just a few minutes to run up the road and get some. Really pleased to welcome Adrienne from Atlanta Georgia.

(APPLAUSE).

ADRIENNE LOWE: Thank you so much for coming to my talk. I am Adrienne, thanks for cutting your lunch short. It is an honour and a great pleasure to be here, like Daniele said, I will tell you about what it is like for a professional cook to get into coding. How cook and coding are not that different.

My goal, if you are new to get started with Django, to tell you a few lessons.

Before I get started I want to say thank you to Jamie and for the foundation that enabled me to come from Atlanta means so much to me that I get to come over here and share a bit of my journey, I am proud to be part of a community that values diverse perspectives, believes some of us who get started with Python and Django, have a unique and meaningful things to say and given a platform to say them.

I hope you that feel you feel the same sense of respect and welcome, if you are coming for the open day. It is something we are working hard to engender, as you are here, please feel invited to get curious to ask us, to share your valuable perspective, we really deeply value the input of the new coder, whether you are here for years you are welcome.

Something I will go into later, it is remarkable how often talking something through with someone new helps me gain a greater sense of the subject at hand. I think this applies to cooking and programming.

Thanks for having me here from Atlanta Georgia. This was supposed to be the point where I mentioned my southern accent but, this morning, on the way here, I got a cab because I was running a little late. I didn't want to miss the first talk. The driver was like, so, cab driver, where are you from? Canada right? Oh I was like oh, I am from Atlanta Georgia, maybe I should give you a sense of what I sounded like when I was small.

So, maybe I will read the PyLadies motto, back home, I am the co-organiser of the....

In my 7 year old accent? Fair. Good.

Our mission is to promote educate and advance a diverse Python Community throughout reach, education, conferences, events and social gathering. (APPLAUSE).

That is really what I sounded like when I was little.

But I did drama in high school in college and ...

NEW SPEAKER: Can you teach us how to do that?

ADRIENNE LOWE: Yes, I did a production of Romeo and Juliet and I had to do a Cockney accent.

So I mentioned back home, co-organiser of a PyLadies HDL and if you don't know what PyLadies is, I will give a quick plug for it, an international mentorship programme. Helps women become leaders in the community.

For the first time we will be Django Girls to Atlanta because of me, there are a lot of people. I am an advocate for the group.

We are here at Djangocon, maybe curious why I am using the word Python., a refresher, Django is a free and open-source. ... programme that is use Django that you may have used already include the social science pinterest and ... ... discuss, we here like Django because it enables us to build applications rapidly, cleanly, securely with the minimum of repetition. Also superfast as the Django Girls workshop proves time and time again. Even someone without experience can build a Blog about a tutorial in a Blog in about a day.

So now that we have got some of the terminology down I want to take a few moments to tell you about myself and share what it is like to come from Django, from an entirely untechnical background. I grew up on a family farm in Georgia where we always had a huge garden this was a source of a lot of contention for my parents because they are about 250 ochre plants in the image, no family of 4 need 250, but my dad thought it was a good idea. That is what you see, 250 plants. We all love food, I started cooking at a pretty young age, went to college for history and philosophy. I started working in kitchens full time, working up to 80 hours a week. I moved to grad school, I went to a Quakers Seminary. Frequently cooked for one hundred person community mill and wrote my Masters on the different interfaith conceptions of hospitality and how different faiths think and share about food.

Grad school was where I used free and open-source software. Our lab was ... which I think was really cool. That is not common at a lot of American Universities, most are Mac everything, so cool we were using to (NAME - INAUDIBLE). Since finishing, I love this image from this year's ... since finishing grad school, worked as a personal chef, doing many consulting for restaurants, lots of restaurant related stuff private and group cooking practices. I came through coding through the passion for open-source software as a user, I want to help women in a nontechnical backgrounds get started on the journey with programming. Along with the rest of the folks hosting this conference I believe that you do not have to have a degree in computer science a background in math or engineering or really any experience coding to get started with Django today. But there are a few things while I have getting started I want to share with you today, might save you a little bit of time.

So first, I want to emphasise how helpful it is to source your learning materials, your tutorials from a wide variety of perspectives and to remember, this is very important, to be very general with yourself if it seems like one tutorial isn't working out for you. When I started learning Python, I did myself a tremendous disservice by using one tutorial and one alone. When that tutorial wasn't really working out for me, I thought, I was the problem and that was very discouraging. I started to doubt my ability and to ask myself if I should even be doing this?

Let's contrast that with the experience of being in the community. So you are learning to cook and you have to use recipes, don't listen to anyone who doesn't tell you they use recipes, anyone who is serious has to use recipes. You want to learn to cook a particular dish. You get a cook book, what happens when you read it, you realise you don't really understand the way the recipe is written or what the authors are saying and in general it doesn't make sense. Probably just get another cook book right? That makes a lot of sense. This bakery book didn't work, so I will try this one instead. That seems like a sensible thing to do, as a new coder, I thought I was the problem. One teacher's method wasn't working for me, and since he is the expert right, like I am must be the problem?

Wrong. Just like you had you may have had a teacher in grade school who you really connected with, who really inspired you, who helped you understand material that you found really difficult. You will find that some resources you connect with better than others. That is totally okay.

You are not wrong. You just need to find out what works and you will. There are really too many great resources outed there for anyone to drag themselves over the coals because one is not working.

You get another cook book. The second suggestion, please let us know if something doesn't make sense and reach out when you are struggling. When I struggled with a particular tutorial, I made a sat tweet on twitter, it was the first time I had ever done anything like that. I was really nervous about it. I didn't want to seem pathetic or didn't know what I was doing. But the truth is, like, I was sad. I was anxious confused, doubted myself. That tweet happened to get noticed by my friend Anna, incredible friend she is an inspiration to me and a leader in our community. Co-organiser of Django Girls and Djangocon, you should follow her, there is her handle.

She is a chapter leader of the new PyLadies remote. Which is something we are working on. Runs a, shares interviews with women whose stories of using Django, sometimes from a nontechnical background, if you want more experience, check that out.

Anna is also self-taught from a nontechnical background. It reminded her of how she felt when she first got started. Wrote me a long thoughtful e-mail, it is okay if one tutorial is not working for you, shared her stories and shared a long list of new resources to try.

So if I hadn't taken to twitter to let off steam, perhaps this interaction never would have happened, our friendship wouldn't have an opportunity to be deepened. Maybe I would have ended my studies.

Come on in, you are welcome here.

Hope lunch was good if you had it.

But I tried certain things, I watched "Amazing Python talk" signed up for a course, already done it through Georgia tech on music technology, we used Python at the end. I didn't think to look into more of Python. She gave me some good suggestions there, I finally started reading, how to think like a computer scientist, I got the help to move forward.

The reason I want to remind you, when you are first starting out to code, it can be really hard to reach out to you the more experienced developers and in general to let people know that you are struggling. Sometimes you feel like the risk of exposing your vulnerability or who you really are much greater than the potential reward.

So, as a new coder you keep to yourself, you say, oh no, no, I am not going the bother my more experienced friend about that because I have received that you are so much smarter than I am. We minimise ourselves, we say, oh this is probably some little thing that I should already know and, oh man, why don't I already know it? Maybe I shouldn't be doing this?

So like the example of hitting a dead end with a particular tutorial, not reaching out, you can feel insular and isolate yourself and it will cause you doubt your potential and purpose. It will keep you getting from what you need to continue on your journey with Django.

So, I want to share that I found that in every case where I have taken a risk and shared when I was struggling this community, you guys, have poured yourself out to me, you have been there for me, you have showed up in incredible ways. It wasn't in my Blog post in coding with knives, that got noticed by Django Europe, finished up in my invitation to speak here, if I hadn't taken the risk, perhaps I wouldn't be here today. I can't think of a more powerful personal example to share where you are and ask for help when you need it. Who knows what marvellous opportunities await you? Please take my word for it, speak up, say what you want to make do, dream, especially when you are struggling.

If you do feel that you are burdening someone by asking questions, the better you get, the better he/she will get and the better altogether. This is because we are asking what feels basic to you, gives someone an opportunity to flex their teaching skills. It is mutual learning, you get the answer and your teacher learns more to hand from their interaction with you.

I, as experienced cook, I am so far removed from the experience of being new, I move too quickly through my explanation, take for granted that someone understands, it is great to ... never feel like a waste of my time. Help me rethink my processes and procedures and help me to be a more effective cook and educator.

My third piece of advice, have a goal for yourself, sometimes this manifests as a project you want to work on or build. It may be a more general goal but, you can use it to refer to and use as guidance during the very inevitable periods of discouragement and de-motivation.

The thing about teaching yourself to code and from what I understand about the experience of programming in general, professional programming in general, is that you are going to have moments when you feel like such a genius, like you are going to do this tutorial and you are like, oh my gosh I got this. But I think, also be in equal measure those times when you feel like you are not getting anything and nothing makes sense and maybe again you will start to doubt yourself.

I just like this. Somebody else used it in a presentation, it resonated with me. You may think your goal looks like that, but in the middle there is tangly bits. Likewise cooking professionally is hard you guys! there are lots of opportunities that through your spoon go in your hair and storm out of the kitchen. I am not sure what the working conditions are in the UK, in the US, kitchen work is notorious for 4 things, long and inflexible hours, bosses, no health benefits and really low pay. So, it is not very appealing work.

Yet, millions of us still do it as servers, line cooks, food runners and yes even chefs. I think it is because whether you cook professionally or for your friends at home. The most satisfying, we simply love seeing the delight on your face when we put something we made down in front of you. I spend way more time cooking for people -- I won't feel my best if I don't serve you a simple meal. So you can prepare yourself for these periods of frustration and encouragement, being mindful of your goal, help guide you through the times when you want to give up.

For me, my goals to continue to open-source projects which I want to talk to you about later and to help get more women into Python and Django, the two projects I use, I feel you have given me so much and I want to give more back, getting more women into computing is precious to me. I had to fight for it myself and I would like to make it easier for women like me.

When I start to feel down and out, I just remember these goals that is kind of like setting a big plate of food in front of somebody I love. Like these lovely people at a dinner party I have.

There you go, examples of my cooking and coding are not that different. If you can cook a simple meal for yourself, you can start with Django today. Sourcing materials from the library, don't take it personally if one doesn't work, you are not the issue, you just need another cook book, there are plenty out there.

Let us know when you are struggling, don't be afraid to reach out for help and to bring your full self to the enterprise.

Third and finally, having a goal and purpose will get you through the tough times.

So that was the recap. This is me, my cat, named after the programming language, I have another cat called pearl.

If you want more pictures of my cats you can follow me on instagram and I am also on twitter, follow me on there. Sometimes I tweet about coding and sometimes I don't. If you are interested in coding with knives, I haven't done much in the last two weeks, it is cool., lots of people referred to and said they have found it really helpful. This is the first of two talks, this was the open talk, geared towards newcomers. Giving another one on Wednesday that will maybe be a bit more salient for experienced developers.

Yes, be in touch and just thanks so much. It is a pleasure to be here. (APPLAUSE).

DANIELE PROCIDA: Thank you very much. So we have got a few minutes for some questions. So, if anybody from the floor has a question for Adrienne?

Funnily enough, Russell.

RUSSELL KEITH-MAGEE: Thanks for talk. You mentioned at the beginning you hit a wall with the tutorial. I was wondering if you could, share which tutorial you hit the wall with, and what the nature of the wall was, what got you over that?

ADRIENNE LOWE: Sure. Best possible question. No offence to other potential questions, one that makes me nervous, I don't want to upset anyone. The tutorial was Python the hard way. I don't know Zed, but I am sure he is a lovely person, but I struggled with his tutorial.

Where did I start to struggle? I think part of, I think part of what caused my struggle really had to do with imposter syndrome. Had been invited, received sponsorship to give my first talk at a conference, in the southern US, and I really worked myself up for that. I felt like I had to complete learn Python the hard way in a certain amount of time. That schedule was ultra-condensed and it didn't work out for me. I think I got to some of the later chapters in the 20's and 30's, he would say like, oh you need to spend a week on this or two weeks on this. Then he would have others, oh you can do this in a day and that actually took me much longer, then the things he said would take much longer, was more condensed for me. So some editorialising in his tutorial. It is hard to teach. Just like we all had our favourite teachers in school, like some you will connect with more than others, he was not my favourite teacher, he has done a lot for all of us. But his tutorial didn't work for me. So the thing that got me over the hump was reaching out. I had been documenting my process of. My friend said, you are flying, you need to slow down. She reached out to me and said that. If you see somebody doing that: Dang girl! Feel free to say Dang girl! It must be what they need to hear? Anything else?

DANIELE PROCIDA: I have a question.

ADRIENNE LOWE: Thank you.

DANIELE PROCIDA: What kind of difference are there in the cultures of the programming communities and the cooking communities that you have encountered?

ADRIENNE LOWE: That is interesting, I would say that among us programmers we tend to be free spirited and we have lots of things we want to work really hard and I think we want to make a difference, I think you also see that in the kitchen, people will consider that a plate, to be creative and work very hard. I will say, I haven't experienced this much in this community, the kitchen is highly regimented. We have the chef who is at the top and then we have the Sous chefs and Salad chefs and food runners you get a sense for that kind of, the way it is very regimented. Very much stay in your place, don't step out of your place and be respectful. While I feel we have respect in this community, I don't think we have the sense of you are this and you can't do that. That is freeing especially coming in from a marginalised group. Does that satisfy?

DANIELE PROCIDA: One more question.

FROM THE FLOOR: Anymore projects that bring your coding and your cooking together?

ADRIENNE LOWE: Still learning, I am hoping I have a couple of little things maybe I could chat with you privately about. I am still new. That is why it is an honour to be here in this community.

DANIELE PROCIDA: Thank you so much. Thank you (APPLAUSE).

One thing I have not mentioned, if you haven't already seen in your conference bag one of the items is a signed numbered, limited edition print by a local artist of the animal wall next to Cardiff castle so we approached Cardiff print workshop, asked them to produce their interpretations of the sculptures on this wall a very famous Cardiff landmark so look in your bag look after the print because it's a lovely piece of work and the artist will be at the conference on Wednesday.

I am very pleased to introduce Árni St Sigurðsson from Iceland who is talking about data driven democracy and his work round the Icelandic general elections. Thank you. {Applause}.

ÁRNI ST SIGURÐSSON: My name is Árni St Sigurðsson and I come from Iceland, I have a Bachelor Degree in computer science from Reykjavik. I have worked at 2 of the 3 largest newspapers in Iceland doing work in general. I am currently working on my own start up doing some consultations on the freelancing. My background is mostly development websites and API using tasty py and I think of myself as a programmer although become more intimate with Bashkel(?) and operations I was comfortable with at the time.

I'm going to tell you one leg of the development story of democracy in Iceland; there has been an initiative during the last several elections to get candidates for elect office to fill out the standard questionnaire and using that data to fit and auto {inaudible}. The last decade has seen turmoil sweep Icelandic politics economic catastrophe rocked the country in 2008 and led to a popular protest which ended with the resignation of the majority and an election that shifted the political landscape severely culminating with historic indictment and conviction of the former Prime Minister for not observing laws on - I don't have a translation for it. It's laws on good government that they're supposed to ensure that ministers do their best.

It would not be an overstatement to say the national psyche has been going through a schism. A change of leadership it's felt is not enough. A democratic {inaudible} would solve the problems. Calls for constitutional reform with answered with Athenian style assembly {inaudible} national registry. The document was to inform a {inaudible} convention to {inaudible} reboot. So this is in Icelandic I couldn't find the graphic in English but this a word map of the document produced from that meeting.

Traditionally we have had 2 sorts of elections in Iceland. {Inaudible} watered down by party affiliation ... oops ... one should not get too technical! ... Yeah so it's usual party affiliation, you mark X for a party and it's a simple sort of election. The constitutional convention also posed to be non-political, not partisan acted, so mostly they didn't interfere, but we had 523 candidates for the 25 positions of the convention and this represents the problem, if there is nothing like the elections they had participated infer for, after complaining of lack of choice for decades too much democracy was perceived as a problem and {inaudible} confusion. In essence it came down to making an awkward choice of up to 25 individuals from the {inaudible} and we just scrolled through the short bio and position on the constitution and image of candidates of those 523 candidates so you can imagine you know about 50 people, about 20 people very well, you are about to select people who are going to lay the foundation for your democracy, you want to make a good choice but 523 candidates, that's a lot of choice.

Opportunity to feed an experiment. Being one of the most popular news web-sites in Iceland known for hard hitting journalism and having no political slant in its coverage made it a good choice to run it you probably fill out like a scales - this is an example question. They come in very informed but usually have an odd number of options, 4 or against the proposition. If your {inaudible} is formed you can measure distances between any 2 profiles by summing-up the difference between each question.

A couple of things help the project along. There was no competition in the space and there was a general consensus politicians should not interfere with anything pertaining to constitutional reform, the size being only {inaudible} made it easy to publish the solution. 4 {inaudible} candidates answered. That's 6 out of 7. That's a very good return I think.

If we thought about averages then you would assume that about 7 of the list of 25 would not have answered the questionnaire but it turned out that there were only 2. So, it would seem to be a good strategy for anyone putting forth their name to actually participate.

Another assumption we can form from this data is that also I ran a comparison what should the - I'm sorry I lost my trail. Best I can give us why people {inaudible} not similar in use is name recognition proved a very strong thing, most of the people elected were very known to most Icelanders. There are sorts of {inaudible} political activism, with wide range of opinions among candidates and most people agreed those elected would represent broadly the will of the electorate. There is a separate story to be told about the aftermath but the short version is the work progressed quite openly with relatively little interference with the political class circulating a document which got positive response. The draft was submitted to government and has been a hostage to political class since showing change does not come without its fair share of work.

{Inaudible} for presidential elections in 2012. It's rather unusual for Icelandic politics to see a sitting president being contested. There are 5 challengers and 2 of them strong contenders. 24 questions focused on the power of {inaudible} ... 160 {inaudible}? Balloted a fair proportion of the population were reached. The results were more predicted but there was still this pattern emerging that people would rather have a vote counted for a winner than {inaudible} purely. There was an anticipation of the 2013 parliamentary elections would be historic, the mixture of apathy hope and fear was typical of conventional media. Among politicians and population. Instead of the usual 6 parties to choose from this election put 13 parties {inaudible} each party has to turn in a list of 1890 sponsors to qualify for each district that's 25,000 politically active citizens out of an electorate of {inaudible} there is a {inaudible} party has to get before any votes count towards a seat in Icelandic government. Any party receiving less than 5 per cent of the vote nationally will not gain a number. This rule has been criticised for being too exclusionary. You generally don't get regional slates unless they're recognised as a splinter from a faction with a national relevance. It poses a challenge to end new party. We saw indication in presidential election numbers that people would rather have a vote count than vote for their political positions. All these issues the importance of an election - this time there will be challenges we have not tackled before. The first statistical problem was {inaudible} - elections in Iceland are rather short compared to US a little longer than in the UK liaising with established political parties would be easier and infrastructure is in place and political operative’s season. A multitude of {inaudible} however could be hard to tackle. With those data we would still need to run a campaign among the candidates to answer the questionnaire during the height of their campaigning when {inaudible} doing other things. We featured 63 questions with the option to write additional comments on each question. We gave the candidates an opportunity to write posts that will be featured on the election front page and answer a more casual set of questions. {Inaudible} we also hosted some of the more recognised candidates in our editorial room making a news item out of who now have been added to the pool of answered questions. We also have battle fact that some of the established politicians refused to take part to you due to the media coverage done by {inaudible} remember hard hitting journalism - that's something that actually worked against us there.

The stark {inaudible} here is the currently governing parties are the ones least likely to participate in new project and the only parties whose leaders abstain. The pilots were the first party to have all candidates supply paper and only new candidates to get members to Parliament; 11 per cent of the population voted for a party without getting a representative. Of those 8.6 per cent vote for parties that have enough national following for election {inaudible} distributing and 5 per cent cap. It's very ironic more democracy led to less democracy.

Instead of discussing the municipal elections of 2014 I want to give a brief overview of my architecture at that time. If you want to replicate a project like this it would be a good place to start. Django {inaudible} usual apps and delivers you can do {inaudible} settings and {inaudible} build {inaudible} mostly from the command. Messiness comes about when apps become independent as they often do in media. I offered to make an app {inaudible} list of questions, questions with attached answers and texts attached to questions. This was necessary to be able to have question span municipalities. {Inaudible} event of having to change question wording and to be able to build a separate representation layer for the questions. The last feature was not used but any bilingual country would need that. Each model contained an app field so separate apps could be built to represent each election. It would be relatively easy to build a new election web app for each election but if I were to take out last run I did I would build the application - building an election app web-site would be a push button affair. We made a single change in algorithm from the year before. One {inaudible} decided to check which party best fitted known nothing approach and filled in all the answers as "don't know". It also worried me that it would be a good strategy for a politician to answer only a handful of questions but how the majority we don't know if they counted for a seal or difference with any voter {inaudible}. I fought for a default distance of one for any don't know answer irrespective of what position the voter held penalising not taking a stance, this resulted in a candidate calling our office furious with a question: how come I don't agree with myself? I had to summon my strangest poker face explaining to him 2 people don't know nothing about something indeterminately far from each other but the closest approximation I could come to without being harsh was to say that they probably don't agree.

This is probably more politics than any of you expected but most of the time it's the subject matter that makes the pack interesting instead of the other way around. Voting in a secular democracy is a {inaudible} voters are being drowned in data. For {inaudible} to empower the citizenry, they need tools to see through the fog. I encourage you to light a lamp and show some of the way.

Thank you. {Applause}.

NEW SPEAKER: Can we say thanks to Arnie and has anyone got any questions about this? Yes?

NEW SPEAKER: You said that the position of the newspaper within Icelandic politics was very neutral. Is that all you used to make sure people could {inaudible} have it in technical aspects...?...audit what you were doing to slim down a bit more?

ÁRNI ST SIGURÐSSON: Well, it's more of a culture thing. At the time at least, this newspaper would have been trusted to not take a stance on politics. People were mostly unhappy with coverage of rape cases where they sometimes named convicted people. There was this big case several years ago where a man living in rural Iceland committed suicide on the day that his name was on the front page, so it's hard hitting in that way. They took a lot of flak or it but kind of vindicated several years later when a lot of men who were boys at the time came forth and confirmed that the guy really was a sleaze ball. We didn't have to worry about people not trusting the result. There were a lot of spin-off sites that were using our data to do other things, to do things like raiding each question for the political compass and then doing a graph of where on the scale each party was.

NEW SPEAKER: So the data was opened is that accurate?

ÁRNI ST SIGURÐSSON: Semi-open. You could actually browse through each candidate's position. We made it in booble(?) as well. For anyone that sheared the test the first couple of days the Facebook sharing actually showed how they answered the test so we quickly quit that but - and I was kind of amazed that the government agency for privacy didn't actually contact us and do something. But that was an honest mistake.

NEW SPEAKER: Anyone else with questions?

NEW SPEAKER: Do you see technology as being a good force for use for creating a digital town hall?

ÁRNI ST SIGURÐSSON: Excuse me for a town hall?

NEW SPEAKER: Yeah a way for the public to directly engage and drive decision making in a democracy?

ÁRNI ST SIGURÐSSON: Yes, yes we need this, we need all sorts of projects on democracy and we need funding for people to actually just go ahead and do these projects and publish them not as a Vassall of some company, not as a Vassall of - not as a wage slip, so that more journalists outfits are going to run things like this. In my opinion technology actually opens a whole new avenue of investigation in democracy. If you think about representative government we're actually using the worst form possible at the moment. We're sending someone to Parliament or any elect office for some amount of time and we have no say in what actually he does. So, we need to be able to make a recall, you know, say obviously this representative is not representing us and I actually have an idea. I mean think about it. Think about elections. Think about Facebook. Why aren't elections Facebook basically? why aren't districts groups of people that collaboratively pick somebody from the group who go to Parliament? If the representative is not delivering on his or her promises, then the group can just recall it. I mean, it's relatively simple to do something like this. You could have a layer. My favourite idea is something I've been calling an emergent congress, where at the first layer, you have to convince - you'd have to sign up at the national registry for the election, I want to be in Parliament. And you have to then - you are randomly assigned into a district and you need maybe about 10 people and you have to convince them that you are supposed to win this round and progressively you let people from each group meet people from other 10 groups where you also have to convince everybody else that you're the guy who goes to the - and then you finally have something like final decision or something that the whole group can then vote between, I like this guy, I like that guy. So, if you couple this with recall option what's the probability somebody is going to get chosen who will not do the will of the people? That's very low. That's actually you would say non-existent because this will presumably be a prestigious position people will want to keep and they're not going to keep it if the ones that elected them aren't - well, if they're going to issue a recall and that's the end of your term because the probability of you maintaining the position amongst the people that sent you there when you didn't do what they wanted you to do is low.

NEW SPEAKER: How do you think this can happen in other countries?

ÁRNI ST SIGURÐSSON: Something like the thing I was doing.

NEW SPEAKER: Yes in Iceland right.

ÁRNI ST SIGURÐSSON: The pact behind this is simple. There are, officially things that are going to bite you, if you do something wrong for example, but basically, the first implementation of this was just a model with each question being a field and form model to collect the research and then it is a matter of having an infrastructure to compare, so, you know, after this talk you should be able to pull together in very simple terms something similar. So if you want to do it locally, give me your e-mail, I will be in contact.

FROM THE FLOOR: Thanks.

FROM THE FLOOR: What about hacking and security, that is usually the main thing when we talk of politics, there seems to be no solution.

ÁRNI ST SIGURÐSSON: So you are worried about some of the spoofing candidate for example?

FROM THE FLOOR: Yes.

ÁRNI ST SIGURÐSSON: Well it is not relevant here, it is relevant in everything we do, we should of course worry about security, we should worry about access controls. But, it is not a part of this discussion. It is a part of what we have to do generically for example, we have a pay, actually we have, I am no longer working at the TV so, we had the pay wall. If the pay wall doesn't work, you know, if people can just smuggle themselves inside the pay wall we have no problem. That is where you saw the security issue in the first place.

FROM THE FLOOR: Special problem with security and boarding, you can't mix security and anonymity. So it is not the same problem as the pay wall. Because the pay wall I can look back at my payment and I can know if there is my payment but, by default, definition, if you make them the voting public.

ÁRNI ST SIGURÐSSON: I am not advocating e-voting. Nothing about this is e-voting, all of this gathering data, filtering data and doing something.

DANIELE PROCIDA: I think we will have to stop there, thank you for your questions, thank you very much. (APPLAUSE).

Our next speaker is Žan.

I am pleased to welcome Žan Anderle who is going the be talking about mistakes and lessons learned, that is a common thing in our field. So. (APPLAUSE).

ŽAN ANDERLE: Thank you for that. So, hello everyone. Before I start I would like to get a feel of who is in the room. So, how many of you here are students?

Okay. A couple. How many of you do back end?

Okay most of you. Front end?

... okay.

Also where you work, does user experience, is user experience something that is taking care of versus something that just happens on the way? So who works in a company or a place where it is taken care of?

Okay some of you. About half.

Okay thank you for that.

This talk is mainly about user experience, it is not specific to Django, where I work we use Django, will be happy to talk about that later but it is not going the be a part of this talk.

My name is Žan, I come from Slovenia, I studied maths science and Bachelor Degree in mechanical engineering, I currently work at datafy.it. I am in charge of user experience.

I don't want to make assumption, let's get that out of the way. One possible definition, it is basically how a person feels when they are interacting with the system. In this talk I will be using the term broadly, like sometimes it is better to use usability or user friendliness but we are not going the dwell on that.

So, say you are a developer who has little to no knowledge about user experience or you work in a team where, a team that doesn't have money or time to devote to development of user experience. What do you do about it?

That is basically what I am here to talk about. It is what happened to us at datafy.it and it is a start-up. We managed to do something about it. Let's have a look at the mistakes made and the lessons learned along the way.

But first let's get some context.

So datafy.it is basically is search engine for business contacts.

You get the country, what you get in return is business contacts. We are four developers and two sales people and because we are a start-up and a small team, this is what begs the talk rather than as we face this, this issues of not being able to devote a lot of resources to user experience.

So before we got the version of the app you saw on the slide before, we had bunch of versions before that. We started with an app in command line and the first public interfacing app was this piece over here. It was very complicated. I had tonnes of features that shouldn't be there, yes, it was really complicated. Complicated to maintain, difficult to explain how it works. So needless to say, the user experience from that first version to today changed drastically.

Basically, at first it was a lot of users were using to ... because it was difficult. For every new users the sales had to go through the process of how it works and explain everything which doesn't really scale because how are you going the sell to a lot of people if you have to explain it to every single one personally?

We got a lot of calls customer call related., we spent a lot of time to explain how everything is working.

Now fast forward to today, the one thing that people point out, time and again is the use of our app. Basically, do no more teaching of how to use it, it is self-explanatory, yes. Any calls or e-mails are just bug related and not user experience, that is good I guess.

Yes. There is still a lot of, a lot of stuff to improve but, we have learned a lot on this way and we have messed up a lot of things and through messing up we have learned a lot of things and this is what I will talk about.

I mean, this talk comes from all the mistakes. A quick think about user experience is, I notice it is, it gets ignored. As developers we seem to forget we are developing for the users. I don't know if that is us being lazy or ignorant but, it is not good because it matters. Since even if your app has killer features, if it is unusable painful to use, people don't understand it, it doesn't matter.

Why is it that we tend to forget about it? I think it comes down three reasons, lack of resources, lack of knowledge and ignorance.

Now through the mistakes and lessons I will talk about the first two, as someone coming from a small team, but just quickly about ignorance. I feel that a lot of times developers see their users as like that. [on screen].

Yes.

I think it is important to realise that when a user struggles with your app. It is your app's fault not user. If you blame the user all the advice I will talk about now, they don't make any sense because, there is an issue elsewhere.

Also I think it is really important to realise that developing pretty good user experience is quite easy. Now, getting to a phenomenal experience that is hard, that is not something we will talk about today.

So finally, this is the crux of the talk. Let's go through the seven mistakes we have made and seven lessons we have learned.

Now some of these may seem obvious and they are after you have learned them but, until you have, they are not obvious.

So as I mentioned before, what happened to us was that as a start-up we have to focus on getting the product out there. We have to focus on so many things and we don't have the time or money to have a single person work solely on user experience. Because of that, what usually happens is, teams don't assign anyone to user experience at all. Because someone can't be there 100% of the time. The thing is, if this is the case, you can't expect results.

For us it was kind of by chance, I can't really retrace how it happened just one day I took over user experience and you should take that with a grain of salt because I still did, I still did and I still do front and back and full stack. I just became the person to talk to about user experience.

Ideally someone would be assigned to user experience, focus all their time in reality there is no way usually and what we have learned is that you get great results regardless. Even if I still worked on other stuff, this proved to be very helpful.

So, in our office when we discussed the features or stuff we want to work on, we have sales people, developers, CEO, there is no user involved in that, in that debate. So when they talk, they talk about okay how we going to solve this from the technical side? What it means about the big picture. All that stuff yes, but no user. That is not really good because users should always be involved in that, those debates. Someone should represent them and someone should be there to defend their point of view. So whatever you are considering a new, a new the feature, new feature, does this make sense for the user, is this user friendly?

This came as a second nature once we assigned someone to be in charge.

As developers I feel we are often content in our own little bubbles, and we don't talk to customers because it isn't our job. Like the boss has moved, we don't talk to customers. When it comes to user experience you need to have input from the actual users. If you don't, you will just guessing and speculating what, what experience is for them.

This for us, this was, I remember one day, I heard my boss talking on a phone with a customer. I heard some complaints. Just by hearing that, I immediately got, I immediately realised, that is not okay, we should change that. It was 5 minutes work, if I didn't hear the input I wouldn't know that. He didn't think of passing that on to the developer’s team. So, I think it is really important to, to include developers to customer support. So a really good advice I have is that all, any feedback you get from customers which is usually forwarded to sales or other, or what is the word? Other parts of the office, developers should be included in that too. What we do is, whenever we get feedback, they toward it to me, so, they forward to me, so I can see if the feedback they gave and what they are struggling with is part of a bigger problem.

This was one of the most beneficial things we did. Through that, realised, error messages were worded in a way that confused the users. We had for example, when you wanted to export the search results you would get a comma separated value format. Which made perfect sense to us as developers like you can import it to anywhere., to every customer that was painful. Didn't know how to convert it to Microsoft excel.

Also we realised for every single user, they have to like as I said, at first we had to explain how to, how to get started with the app. I simply created an intro to our app, which took less than a day, now everyone just gets it which is awesome.

Now, complicating is always an issue because especially for the developers I think we always have, we are working on something and we always have ten different ideas what else we could, ten new features, ten more ways how to improve something.

That is not always good because then you end up with, with so many features that it is difficult for users. It is difficult for you as a developer to maintain everything, basically nobody has a good time.

For example, we had a really complicated export which contained all of the data we could get. Which sounds like a good idea right? Just give them everything but, in the end it just confused the users.

Also we would offer pausing and stopping of the searching and scheduling and all kinds of stuff which you don't really need for a search engine.

So, simplify.

You have to be very critical of what features you are going the add and keep. Basically, in the office we have a rule that we are not going, we are always going the assume that no one is going the need a certain feature and move forward only if we have evidence that it is something that is going the, some kind of use to someone.

Now this one is tricky. Use of technical language. We don't even, we are not even aware that we are doing it but let's, I will just give you an example. So, we are doing a search engine and at first when the button or like start search we call it "start query," when we mentioned a search, we said query. Of course it is a query, why would that be complicated? Almost no one understood what we meant by a query. So simply by changing that to search, it made things a lot easier for everyone.

So yes, try to use plain English, it is so hard, since we are involved in that language, we are deaf to it, we don't realise the word might not be natural to someone else.

Yes. So, not listening to users. Listening to users means many different things and it means listening to them when they give feedback but also means testing new features with actual users. Sitting down them next to a computer, seeing how they interact with what you design. In ideal world you would do user testing but this would be very extensive you would have to get different people in the office and in reality you can't always do that, but we have learned that if you bring someone from the next office or someone from just the different person doesn't have to be an actual person and get them to use your app and observe it, you get so much information from it, it is incredible. I am surprised every time.

So always test new features with actual users. With actual people.

Finally, sometimes it is just failing to adopt basic principles. So, in user experience there are like ten basic principles if you follow just two and follow them consistently, you can get better user experience of the app than most other apps.

It is kind of surprising how easy it is to ignore this or forget about this. But yes, consistency and feedback meaning consistency, do your buttons in your web app always represent the same thing? Is a link that is pointed out wards, is it always styled the same way? Whatever action a user might take, do they get a feedback for it?

So on.

For example, with us, it was the first version was when they started the search it took like a minute to get started and in that minute nothing happened so usually just I mean obviously like, if I was there, I would do the same, click until something happens right?

It is really frustrating but, just a simple tweak was that, so yes. This was what happened after you read a search. Nothing. Even like, even after a, it went after last ten searches it was easy to miss.

Okay. Now you get a notification and it is separated from last searches so it is clear you started a search. It is a simple thing but so easy to miss.

So basically starting taking care of user experience because it is not that hard. Regardless of your knowledge, it is simple things you just have to be consistent and critical of yourself.

As I said. That is it. Thank you.

(APPLAUSE).

DANIELE PROCIDA: Thank you very much. We have got time for a couple of pretty short questions and short answers I guess.

FROM THE FLOOR: Hi, you mentioned that you assumed that you don't need new features or your users doesn't need new features unless you got evidence. What evidence do you need?

ŽAN ANDERLE: It can be many things for example, getting a person to request it. So getting a person to request it and say they are willing to pay for it. So sometimes you often like we will create a landing page, selling something, and we haven't even developed yet, but to see if there is any interest, if there is, we will develop it and otherwise probably not.

NEW SPEAKER: My question is what about why first and did technology you use have or doesn't have to do {inaudible}.

ŽAN ANDERLE: Sorry can you repeat that?

NEW SPEAKER: My question was about the mobile first applicability to such experience and if the general technology you use helps or make it more difficult to implement mobile first.

ŽAN ANDERLE: Yes from mobile first design we use boot strap from everything we doth if you use boot strap properly it kind of forces you to do mobile first. That's been working great for us.

NEW SPEAKER: One more up there.

NEW SPEAKER: How do you measure good user experience like for instance if you are a data driven team and implemented something and wanted to measure the effect you have any advice on that?

ŽAN ANDERLE: I think that is one of the controversial topics when it comes to user experience because it is very hard to measure. For us it was very obvious by the amount of user support related questions we got. So before even though we had a lot less customers we were dealing with customer support, support all the time, things they didn't understand, things we had to explain, that kind of stuff, and now we have a lot more customers and that just simply does not come up and that can be one indicator that user experience is better but other than that it's, yes, through user testing so when you sit down with a user to see how they use your app you get to see where they struggle and that can also be one way to tell if it's good or not.

DANIELE PROCIDA: Thank you very much. {Applause} next stop Alasdair Nicol on ponies and moustaches which are in fact technical terms in Django. {Applause}.

ALASDAIR NICOL: Good afternoon and I am going to talk about ponies and moustaches the more silver templates. I'm Alasdair Nicol a developer at Memset hosting concern south-east England. Started using Django in 2009. Like Adrienne who had the slot this afternoon the first time I used Django tutorial it didn't work at all and I came back a bit later and it worked the second time so it's good advice to try again if it doesn't work first.

In 2009 Django 1.0 came out, we've seen a few releases since then and in April 1, Django 1.8 came out. It's a long-term support release so if you upgrade to it you get security releases for quite sometime and some of the highlights API for the model meta class, there are lots of goodies if you use the postgres database and the thing I'm interested in talking about support for multiple template engines.

So, for any visitors who come on to the open day and don't know what a template engine is I'll start simply. A template is just a text file with a special - and the template engine can take advantage of special syntax in the template so in this case we've got "hello my name is" then 2 variables name and subject. Then if we have a context so the variables, name and subject, then a template engine can render a template and merge the context into the template. So even if you haven't used Django before you might have used mail merge in office or you might have seen an email where you got some these current in braces which means someone has messed up somewhere.

Before Django 1.8 if you wanted to - your choice of template engines there was a single choice of template angles Django template language. This is perhaps a bit surprising there was only one choice - not surprising there was only one choice when you get other parts of the Django stack there has been lots of choices for some time. Mark this morning was saying that with Django you have a choice of operating systems, choice of servers you employ Django with then what I'm looking at is within Django there is different layers and you have been able to have different choices before so when you come to Django and are choosing database you had lots of choices in database. Postgres SQL very popular with distributors in Django, my SQL which is very ubiquitous what we use, SQ light which is built into python which means you can get up and running very quickly, oracle and there is third party databases you can use as well.

Similarly sessions the cookies or the other way you store data about the user accessing your web-site, with Django you can store sessions in the database, use signed cookies or store on files though probably not a great idea to do that.

Finally one more example part of the Django stack swappable components caching where you take a page or part of your page and save it in memory so to speed up your web-site. There is mem cached support built into Django and third party support for Reddis or you can do caching in other ways as well.

Templates in Django - before Django 1.8 there was only one template language you could use. There were occasionally proposals Django should replace the Django template language with something like jinja. I wouldn't have liked that because we have hundreds of templates that work and if I had to change them to something else it wouldn't be fun.

It was a crowd funded campaign launched by Americ Augustin, it's one of a few Django features which has used Crowd funding and indigo {inaudible} seems to be working well. Any way the 3 planks of the campaign or the 3 goals were to keep support for the existing Django template language add support for jinja 2 and API so other people could swap in any template language they wanted.

So for the rest of this talk I'm going to have an example web page which I'll show you in a second and go through the 3 bullet points to see how we implement with Django template language with jinja 2 and another engine.

Sorry this is so basic, my skills are round the back end rather than front end development but since Djangocon here is a table how we moved from 2009 in Prague to 2015 here today in Cardiff.

Django is model template view or also other frameworks use module view controller so before we get to the template we have a model which describes the database layout in Python code. Got a very simple conference model. Going to store the location and country as Charfields {inaudible} country would be better. Years into the field. The view is where you describe what data you want to display on the page resident Python. A very simple view which grabs conferences from the database then renders the template. And finally here is the template to - or section of that template in the Django template language. So, we've got conference year, conference location, conference country, the double angle braces or moustaches, including variables and then the other couple of things this template has is we've got a forward, that's using a template tag where you've got the angle of brace and percentage and we've also - we're using the cycle tag so that we can display {inaudible} even as odd not shown very well on the slide but you can imagine there is {inaudible} extracts going through that column.

So, moving on to jinja 2 which in Django 1.8 there is support for out of the box, jinja 2 syntax is very similar to Django's, very fast. Used by flask, a micro framework in Python and I have been exploring Ansible recently and it uses jinja 2 for its template as well. Enabling jinja 2 in 1.8 is easy. There is a new templates list and by default it will come up with one entry, the Django template engine and you just need to add the second dictionary to enable jinja 2. Then when you write your jinja templates you put them in a directory with jinja 2 and when you use the short cuts Django will take place of everything else. So here is our template in jinja 2 which works very, very similar to our original template in Django template language. I think the only difference in this template is that instead of using a cycle template tag we've moved our cycle instead and jinja allows you to cull functions within the language which Django doesn't. I think that's a philosophical language, {inaudible} thinks you should keep language out of templates and jinja allows more logic into templates.

Now on to the third part - I had a go at seeing what the API was - see what the API was like to add support for another templating language and I chose moustache so I could use the title I talked about. The other reason I thought moustache was interesting was that there are so many languages that have support for moustache and we've had couple of talks today where people talked about rest APIs and perhaps the need for doing templating on the server side isn't as much as there used to be but maybe there is a future where you're doing templating on the server side and on the browser in which case something like moustache where support for Java script and...(inaudible)...is potentially interesting.

Yes in Python there is a module pystache which {inaudible} I used.

Here is my template engine. You need a template object which knows how to render itself and most of that was taken care of using pystache then you subclass {inaudible} base engine. You set what you want to keep {inaudible} moustache directory and then we implement 2 methods, one which will render string and the other render template style from directory and render it moustache template engine.

Here is a template written in moustache. Again using double angular braces for variables. Looking through conferences looks a bit \*different. There is not a 4 conference and conferences syntax like we had before. And the one thing you might notice is that I couldn't work out how to - I couldn't find an equivalent for the cycle tag in Django template language although you couldn't see it on the slide any way so never matter and you could do it with CSS selecting even {inaudible} any way.

So, we've got the 3 layers, the model, the template and the view, and in order to get the view to work with moustache had to change it slightly and add dictionaries instead of conference who ran objects to whatever was rendering the template so I cheated slightly but I think that I have shown them that you can for the most part keep your models and views as they are and swap in whatever template language you want.

So, I am going to declare success. My pony has grown a moustache. If you want any further reading, I am not a {inaudible} don't worry. The weekly updates from Amaric about decisions he made and process are interesting it's Django open source and it's great you can see the collaboration decisions and?? Getting contributed. There is a new process called Django DEPs similar to Python DEPs and I think this multiple template engine is the only DEP so far that's been accepted but it's interesting again to see a new way of how decisions about the future of Django are made in public and by the community. The Django design philosophy page on the web-site says a little bit about why Django thinks there should be less - about some of the decisions made about Django template language, I touched on that briefly when I said there is not as much logic in the Django templates as the jinja templates and there is 2 scoops of Django which is a really great book and the new edition has a chapter on jinja and some good hits if you were going to be seriously considering using it in production rather than just a fine exercise.

So, yes that's my slot. Thank you very much. {Applause}.

DANIELE PROCIDA: Thank you very much. Not least because your helpless claw back lost time. A coffee break waiting for us. Thank you very much. I already saw a couple of hands moving in the audience so who would like to go first with a question? OK I'll start. So, why might one not want to go down this route of exploring additional templates? Why not just stick with the completely built in out the box Django template language that everyone else uses that you know has been widely tested and will do pretty much every tutorial documentation at work? {Laughter}.

ALASDAIR NICOL: I think that's a very good point and I personally work anyway - I don't think we're going to be switching any time soon. If you had a performance reason you might want to switch to jinja although many people say there is other areas like caching or database where you get more performance rather than templates any way so I like the Django template language, I'm not advocating we switch to something else but I think it's really good we formalise the API and given people the ability to swap it out if they want to and I think there is some of the even by making the API more generic it's improved the way the Django template language itself interfaces to me to some extent -

DANIELE PROCIDA: It makes me nervous when I feel there is an implication I might be expected to learn something new.

ALASDAIR NICOL: I agree like a couple of talks today suggested that the future is REST APIs but actually it is very nice that with the Django template language at the moment you can get your tutorial or your log in the Django girls tutorial up and running and you don't need to learn yet another front end single page app.

NEW SPEAKER: By reaching engine {inaudible} template packs?

ALASDAIR NICOL: Yes I think that is probably the biggest - so the templates act yes I think you'll need to re-write right the templates act. The filters - I think say if you use jinja, I think you can register filters - a filter is just a python {inaudible} takes a value and one other and returns another value. I think in jinja 2 you can put filters but tabs if you switch to jinja that involves quite a bit of re-write.

NEW SPEAKER: Is there currently the ability to choose a template to run in {inaudible} time so if you wanted to render may be an Ajax view slightly differently with a different engine {inaudible}.

ALASDAIR NICOL: Yes so the example that I used was you do - templates by configuration so you stick your jinja templates in jinja directorate and Django will look for them but when you use rendering in Django 1.8 it's got engine key word so you can have - I have got example template that can do most of this stuff and the way I've shown the view 3 different times is with a get parameter which the engine equals Django or engine equals moustache gets passed through but yes at one time you can choose engine.

DANIELE PROCIDA: Very quick one.

NEW SPEAKER: If I understand correctly the reason the logic is taken away out of Django template languages is so you get better design principles. If you follow jinja is that taken down a darker path? {Laughter}.

ALASDAIR NICOL: Ah ... so I think perhaps you'd be in danger of going down that dark path because one thing jinja has that I would like Django template language which is to be able to look up and attribute by variable which you can't do, you can use it basically it looks like dictionary look up syntax in jinja which you can't do in Django I think that would be nice in Django, I don't think the world would implode if we did that. I think the Django philosophy of trying to keep logic out of templates I side with it to an extent.

DANIELE PROCIDA: Smashing. Thank you very much. {Applause} we have a coffee break now, so go and have some refreshments. We'll come back, try and be here for in 25 minutes time so at 10 past.

(Break)

Hello and welcome to the final session of talks today, we are going the kick off with Christopher Hunt, the lead developer, research assistant and ... at Plymouth university, he will be talking about Arduino sensors.

CHRISTOPHER HUNT: Are we all caffeinated ready for the last session, tried to Welsh cakes? Hopefully you have.

So welcome, today I am talking about we are going to talk quickly and speed through, try and scare these people doing a lovely job in the corner here.

So, come in people, you have missed the start.

So the Django and the data system. Objectives for today, understand the concept and come points of the data system, a metaphor for the things I have been using, the different processes and systems.

Discover we, i-DAT is using Django, try and provide two live demo, only one now, because some of these didn't survive the train journey up here.

I am from institute of digital art and technology, we are an open research lab for playful experimentation with creative technology. Very fancy. I didn't come up with that, my boss did.

We exist at Plymouth and support the digital art and technology undergrad course, research, environmental things, social stuff, domes and different environments, biological and lots of other key things.

Key line from our description of our strands tat bottom. From data to code to experience to behaviour.

This is the process and this is what the systems and what we can do enables. So as we go through, keep that in mind and keep thinking, what kind of things, what stuff you can enable, tools and technologies we can play with as we go through this stuff.

So, the data system. This is my model with lovely ASCII art I am proud of this. Start at the top. Circular system because it goes around in circles.

At the top we have sense and collect. This is sensors, these are these devices, things that you know, like sensors, a few other things, they transmit to somewhere we can store and process that data. This is where Django comes in and that is where I will talk about that. Then that enables interpretation and understanding of that data. This is visualisation, this is oculus rifts, and cool stuff at the end.

Hopefully through all of it. So first step. Sense and collect. Which transmits to somewhere we store and process.

In here, this set of transactional props comes from PhD projects this is done with myself and one of my colleagues, John and in here, we have these tiny little formally sparkled now particle. So I have got the re-wipe that bit of code in my head. Renaming things it is annoying.

These are little Arduino light devices, programme them using the same languages, wired directly into the device.

So, what is going to show that but because, in the interest of time and with a few network issues, what happens with this project two people sit down. Then, these devices all have different sensors the funnel has an fsr, you squeeze and then it, certain cups vibrate and little nozzle tips light up and do things. The idea you are using the sensors and exploring the relationship and the system beneath. So behind this, there is this Django page which is receiving all the data over something called message cue ... transport.

All these devices sending up to here, we can see this going on, this goes to another back end, we can explore and see the data then different stuff happens there., I may need to ask whether somebody has a soldering iron.

Pretty standard code, except extra bits for connecting to a broker, connecting to NQTP. It is a publishing subscribe protocol, nice and easy to connect with., here we are connecting to our NQTP broker, then publishing our information our messages, so, the green glass, what happens here in its loop, checks if it is connected. Every time I tap that, that would then send that message, that fault it received over NQTP then make it vibrate, but we will show you that later.

So that is sensor, those are Arduino's and different devices but, at the essence of that, that is just data, data can be any kind of thing, we have done a lot of project and research with arts organisations around different, about how you gain and manage feedback. Traditionally someone would pull, go around an art gallery, someone pull you aside with a clipboard to answer lots of boring questions. What comes to your mind when you think of science?

That is, then you have got this open ended, this event did not match my expectations dry and not interesting boring things. You start to see lots of ways to get around this. This is a happy or not sensor, the question here: Please rate our check outs.

You may have seen these at airports as you go through the security gates. You go through, then slam on the button say, hey I enjoyed it or this went badly.

I can imagine sometime next week you might see something like this, dotted around Cardiff! just in case you need to voted on Zain, say you are a one direction manager, find out who next to put on the bottle of fragrance, we will get back to that later.

This is developed into our quali project, this is where the mobile apps come in. This is the same kind of thing, an app for the Cheltenham Festivals. As you go around the festival, as you explore and find out stuff about it you get to rate and leave feedback. Say what mood you are in or did you enjoy the event. In the micro-interactions, instead of having a big long essay of things to fill out. Tiny bits of data, as we collect more and more of that, we build up a better picture of what is going on.

So, devices, sensors, mobile apps all cool. They all transmit back to somewhere to store and process the information then enables interpretation and understanding.

So, we could use fi base or pass or all the platforms of the service thing, we are at Djangocon, I would encourage you, why not build your own? A good question. You could save time with those, but you want to learn and, boar reed this quote from my boss's talk mike. I will give it my best Tom Hardy impersonation.

"take control of your city. This is the instrument of your liberation!"

I really should have been an actor not a developer clearly.

But that is it. It is for us, it is being able to control a platform rather than rely on someone else or worry about things degrading, things being unable and falling apart so, patch ... cosm and xively, lots of stuff out in the field broke. Great. Amazing that.

So you learn and that ability to control and build stuff really is valuable. This is an old screen shot from digital ocean, but the cost of computation, the cost of running the server is so small you know, you can easily do quite a lot of big project work on just a tiny VPS, easy peasy, -- got 5 minutes left.

Dive through the last few bits, why Django? Why use it? Because it is awesome. Why else? Obviously you need a few more reasons than that.

So top of my list, models and migrations, that makes it easier to prototype the data. Here is my model our happy or not button, ignore the code, only for demos, I have forgotten to remove a certain someone from the member choices, that is depressing, the value of Django, that makes it easy for us to prototype, we can iterate, get stuff out. Get people playing with stuff.

REST API libraries, allows us to enable different things so we can talk to tasty py and the rest, there is the data from earlier, of course you have got Python. The only way to describe the power of that, well with that. You have got all the sorts of things, as you have been listening today, you have got numpy, scipi all sorts, takes the work away from you, you can get on with doing amazing cool things. Of course, the kit most important thing in Django, is the community, is you lot. Everyone who is really friendly all the core developers, or core committers are so friendly and helpful. That makes a big difference when we are teaching and working with people, then aren't afraid to get involved, mix in, ask questions, that is brilliant.

So lastly, so we have stored it somewhere, built our Django back end, now we need to interpret and understand the data. Because we are working with arts organisations we can do something like this, just the templating language, so build basic views on demographics, feedback, we can look at hot spots and send Norwegian puppeteers where people are building the apps. Create systems to create questions, get interested and involved in the data we have the immersive vision theatre which is our planetarium, here we can fly you to the edge of space or through the bosses colon, which is the weirdest experience I don't want to experience again!

Or take you into data scapes. So this is something we have been developing for one of our conferences and lastly, I can dive straight into another demo.

Fingers crossed. A ha.

Fantastic.

So, interesting effects, so in here, I can now can't see you obviously. You are all just floating bits of data. But, if I look up there is some pictures been down loaded through one of our back ends, here some blobs of data, bits of orange there. You can come along and experience that later.

But that is all about engaging and getting people involved in the interpretation and the exploring what is available in that data.

Okay. How much longer have I got?

One minute.

Cool. Fantastic.

That means I can do the last few examples, from i-DAT's work, we have a building, we have sensors in a building including a vision system. Can you see this thing moving? Should now start going backwards because the video is looping backwards. So this is our Slofbot, using the whole model. So the sensor is at the top of the atrium, that is looking for detecting people going across the room, as you go along to your lecture, there is suddenly a wall in the way. The idea of that is to kind of get you thinking about the space because you don't think about this corridor, then something is there, I must realise what is going on.

Or, this is very old video now, but or you, or you will look at how people use the space and the building. Then you decide as a long term intervention you put a random duff button in the lift, to get people exploring through the data.

That is the data system, that is all the elements, I will open up to questions before I get pulled off the stage, thank you very much. (APPLAUSE).

NEW SPEAKER: Can I take any, Chris take any questions?

So you mentioned briefly at the start of your talk, how is your work explaining -- do you want to explain?

CHRISTOPHER HUNT: So, the internet thing was, when you say interneter things, you think a smart house, smart car, smart things going on. I don't like using that term, so that is why I use the models with the students and the people I work with, to break down the system going on. It isn't just remote controlling something, but it is interesting things going on with that data.

FROM THE FLOOR: So obviously Python and Django on the server side, but the nasty language in the ... side of things, is there a way of fixing that?

CHRISTOPHER HUNT: That is a good question. There has been some attempts at porting, having Python then generate that C code for you? But, unfortunately, you can't just run Python on these sadly.

RUSSELL KEITH-MAGEE: Is it a hard way limitation?

CHRISTOPHER HUNT: It is a so, on here, what that C code is done, compiled down and then run on a devices. What these devices are, they are essentially dumb. All these are doing is sending their data up to the Django server and then waiting for messages to come back. So we can add all the intelligence, all the interactivity server side and manipulate that and change that without having to change the code on these devices.

FROM THE FLOOR: There are slightly more powerful devices that can run.

CHRISTOPHER HUNT: Yes, these are simple ones here, raspberry pies or eagle bones, it is all becomes enabled and available.

FROM THE FLOOR: How did you make Django talk in PDP?

CHRISTOPHER HUNT: Django is using the pay, ... library, then a written a management command which is supported by supervisor, then runs in the background, listening for messages coming in, validating them and then to the models. If you are interested I can show you the core stuff that is going on.

NEW SPEAKER: I think we have time for one more quick question? If not, then we can thank Christopher Hunt again.

(APPLAUSE).

NEW SPEAKER: While the next talk is getting set up, there has been a slight alteration to the time for tonight’s meal, aiming to be at the restaurant for quarter past 7 now.

NEW SPEAKER: Could you repeat that?

NEW SPEAKER: Change in the time for the meal, aiming for quarter past 7 now.

NEW SPEAKER: So the second speaker of today's section is Katharine Jarmul and works of customer data analysis using Python and today is talking about data wrangling impact.

KATHARINE JARMUL: Hello everybody, nice to be here, I have prepared a talk that is probably after seeing the last hour or so talks a little bit novice for this audience, however hopefully you can use something from it and if not then feel free to use my slides and tell your aunt or cousin or mother how to do data analysis with Python and yes if you want to talk more advanced we can hopefully chat later.

I am Katharine Jarmul, I am at Kjam at most type of tech things; I was originally from Los Angeles, I live in Berlin. I hope everyone is familiar with Pyladies? Yeah? So the original chapter was in Los Angeles and it's really exciting to see now how much it's grown, it's really amazing and makes me feel very warm about the Python community. I've been coding Python since 2008, I started with Django at the Washington post when there was still ugly Adrian hats in the {inaudible} it's a good time yeah we can talk about some of the caching and I am self and mentor taught so I really hope for those of you who are new to Python that you can find and make some connections here today and throughout this next week or that you've already made those because I definitely wouldn't be where I am today if it wasn't for the mentors that helped me.

So, a little intro to what exactly I mean by data wrangling.

So, it's basically the ability to analyse as something with data. Everybody here probably has done way more advanced data analysis than this but the really good news is that you can run any type of reports using Python and data analysis so some of what I do currently is running marketing reports and user analysis, site visitor analysis but you can also do sports analysis with statistics and {inaudible} quite a large open data {inaudible} a great tool to use.

Why use Python it's a scripting language with some real power. There is advanced scientific stack of course which I hope some of us are familiar with. This is a really friendly community. It's really easy to ask questions and get help. And of course it is named after Monty Python and here in the UK that makes it obviously superior to all other languages.

So why do I even care about Python or data analysis? So the way I approach it especially when I'm talking with people who don't know at all about code is do you ever have wrote boring awful tasks and would you rather never do them again? So learn Python and that's what I do. Sadly cannot fold your laundry yet but talk about that sloth balling where do you go, yeah, we'll see if we can get that working. So again Python allows you to have statistical power without necessarily becoming a statistician. You can easily automate things and you can never ever use excel again and hopefully some of us recognise a nice Welsh dragon burning Microsoft excel!

So the first step of any data wrangling is getting hold of data that's interesting for you to use. You probably already have some. There is probably something you do regularly whether it's logging into a utility bills or whatever it might be - you can use that data, another great source is the very large scale open data movement, so data gov UK has quite a lot of good data sets and of course Python supports any number of formats, CD CSV, Executive, PDF, XML, Json, Google docs and I'm sure many of you are aware of supported natively. For those not supported natively there are lots of really useful tools. I recently had the pleasure of using G spread which is a Google spread sheet user which is not the Google spread sheet reader API and it's quite intuitive and easy to use, XLRD is my favourite if you have to output excel text for people in your company and PDF miner has been something I've been recently working with to mine PDF documents.

So data basing. For a quick start especially if let's say you're working with people on your team that don't know SQL yet I would recommend data set, it's developed by a guy named Puto. It's not his name but his handle so to speak and he's part of the news foundation and lives in Berlin and we're grabbing beers next week I'm helping work on some of the bugs so if you have any things on that send me stuff - there is also relational databases, {inaudible} et cetera and non-relational databases so you can use mongo, couch ...

So APIs application programming interfaces are a great place to get data and you can get Python direct with twitter, Instagram, Facebook and tons of other data sets. Let's have a look. Yesterday I had the pleasure of sitting in Cardiff and scrolling through twitter and I wanted to search for {inaudible} did I butcher that - who speaks Welsh? Caerdydd ... if my family were here they would be very not proud of me. And I just wanted to see what people were tweeting about, a gorgeous day yesterday in Cardiff so you can see that we have just some tweets that we were able to get about 10 lines of Python code.

Other APIs that you can use, Google analytics. Google adwords, these are things that are really useful if your customer data that you need to integrate say with a back end so say you need to pull all this analytic data in integrated with the back end and say how long are return customers spending on site etc. that can give you more information. Also, plenty of open government APIs, I've actually been interfacing quite recently with a lot of the open Africa data looking at some of the conflict mining stories there so there is plenty of data sets there, translation APIs again if you need to do anything like that, stock market APIs recipe APIs, million APIs to interface with.

So a little bit on web scraping, if there is not an API and you need to access the data you can build your own API so to speak with a web scraper so Python is uniquely situated in that it is a scripting language and therefore gives you really easy access to read something like HTML H or X document and then allows you to use the information really quickly with the analysis so I find it's uniquely situated. I don't think that's something necessarily that the community has yet so that's really nice reason to know Python.

So if you want to take a peek at a page I recommend LXML it's tremendously fast and has great syntax. If you actually want to click around and use things you can of course use selenium but something recently {inaudible} interact with ghost driver which is a little bit better I think in some ways and then if you really, really need the whole site you can use something like scraping, scraping is tremendously fast and useful and worked on with a great team of developers.

OK so I took some time to just scrape the Django talks for this programme to scrape the content for simple stuff and returns what's on the page {inaudible} - OK big deal everybody here can read a web-site or use a browser that Laos them to read a web-site so imagine if you couldn't so for example right now I'm using PowerPoint because I don't have internet connection so if you don't have internet connection, if you are travelling, if it can't be retranslated by Google translate or if you wanted to run data analysis on it this is a good reason to scrape data off the web. I did data analysis on the talks most common words are "the" and "and", I don't know what's up with these words and we can start to see Django "discusses, describes" and there is a lot of "nice, useful" depending on how interested you are in natural language processing there is a lot of useful APIs out there for stripping out things like these site words, occurrence words; average character length for titles is 35 characters and 7.4 per cent of talks mention Cardiff.

So, another thing that Python allows is allows support for "big data" - I don't really know many people who do require big data but if you do require it you can integrate it with hadoop and pandas numpy are some of my best {inaudible} and anyone not familiar with them it's great to get Wes Anderson's book and start working through some of the examples using panda's you can automate reporting again an allow you to generate reports on the fly or allow you to create normalised generated reports that run every week and allow you to hopefully move some of that front work off your task and then you can run statistical functions, generate graph charts find and remove if you need to outliers or find news outliers, normalise data and perform data clean up so quite a lot of clean up libraries I highly recommend if you haven't used it before taking a look at fuzzy wuzzy is one of my favourites for if you have to do some language processing and it's not always the cleanest.

OK and then of course visualization which is the point so you've ran all of these things together, the data that you need, you've done some statistical analysis and now you're moving on to visualization, Python has great ones, Bokeh is one I have been playing with, matplotlib is standard and pygal has pretty cool SGE related ones and then the ability to easily share coded charts with iPython notebooks so a client I work with currently there is a lot of non-technical people at the company but they can easily run the code that generates the reports they need they can down load the excel document directly from the notebook and I recommend it for teams that need to interface with teams that are may be a little scared to code Python and I found that over time they become a little bit more accepting to play around with it if you just change the variable we can run it for a different data set and that's been kind of exciting.

This is bokeh. If you haven't played with it. Pretty cool visualization just from the grailer(?).

So if you want to know more I know this one lady who writes about Python so have a book coming out with another one of the folks that I worked with at the Washington post it's an O'Riley book and yes if you get it today there is a free Brains pint in it for you!

OK so you can ask me questions now or you can ask me questions later, so I will be here sadly until Wednesday when I have to get back because I have an intensive German course in Berlin but yeah feel free to reach out and thanks so much for listening.

{Applause}.

DANIELE PROCIDA: Thank you very much. So, do we have any questions from our audience?

NEW SPEAKER: What is the hardest data process you've probably faced?

KATHARINE JARMUL: 100 per cent data clean up it's the biggest pain I don't think anyone really likes it but I think that there is quite a lot of powerful tools down for it but I find it to be still the most manual of processes. If it's clean data and I can import it from a database or a clean source that's great I can immediately start using pandas or whatever I feel like using that day but yeah clean up particularly when there is no normalisation of data say like a matching non normalised strings things like that it is just kind of one of those - so may be one day we'll solve that problem, I don't know how but have a pint of Brains and talk about it.

NEW SPEAKER: Do you have a favourite toolkit you use?

KATHARINE JARMUL: Yes I mean the NLTP toolkit the standard one is the one I've most played around with but I think there probably will eventually be one that's maybe somewhere in-between that having to have that entire stack and having to learn so much about NLTP but allowing people to kind of use some of those tools within just a small library, I think fuzzy wuzzy is useful for using that talk analysis.

NEW SPEAKER: Do you know of any way to get data from say film files?

KATHARINE JARMUL: From film files?

NEW SPEAKER: Yes it's made of audio then many frames -

KATHARINE JARMUL: Does anybody have any ideas? I haven't worked with film before.

NEW SPEAKER: FF {inaudible}.

KATHARINE JARMUL: ?? ... Yeah?

NEW SPEAKER: ... Testing your analysis against a known data set? Making sure you run the same {inaudible} reproduces design output always.

KATHARINE JARMUL: That's pretty essential. I think one of the problems that you run into with this and some of why it can't always be tested is that you have to take into account say your handling a standard deviations or out liars right and do you have a normalised or non-normalised data set and that's one of the hardest things is figuring out OK identifying is your data set normally distributed or not and maybe taking different paths depending on that so I think like testing your data a little bit first and getting toe it is an essential first step before you decide OK this is the report I can use with it. If not you are going to find your reports become really skewed because of one particular outlier or a few outliers, I think that's essential and needs to be done more often and I think that determining different pathway depending on data distribution is another key part of that. Thanks so much. {Applause}.

DANIELE PROCIDA: Thank you.

Okay a little announcement, two little announcements whatever you have heard from our website or written down in your programme booklet or in a hand-out or anything else, I am telling you now the correct time for being at dinner tonight. Ignore anything else, aim to be at whichever venue whatever it is, if you have a ticket for the Vegetarian Studio or the Clink at 7:15. From here it takes about, you can amble to either of those destinations in about 20 minutes. The aim to start eating by 7:30, if we aim to be there by 7:15 that will be helpful. If there are sponsors who would like to get their stuff moved to City Hall ready for tomorrow, we will put it in the van and take it down to City Hall which is a short while.

ADRIENNE LOWE: If we purchase hello web app book, Tracey Osborne’s book, deliver it to the registration desk.

FROM THE FLOOR: When does the ...

DANIELE PROCIDA: We will put a board up for you to sign tomorrow. There was another thing to say, it slipped my mind, I am sure it was really important.

Oh if anybody fancies has a spare pair of hands to help put any of the conference stuff into the back of the van when the talks are over, that will be very handy.

Ah yes, so you should already have your tickets, should have a printed out ticket, don't worry if you don't have the printed out ticket, but either have purchased the ticket or had the ticket from us in one form or another. If that is not the case, and you expected to be at one of the restaurants, see me up in my quality room office thing. You can still buy tickets for the VFS they are £15 for a good vegetarian meal., all the Django Girls ... you will have a chance to go the different restaurants.

So, yes?

So, you have got 20 minutes including questions. So, Rivo Laks travelled from Estonia, on the real-time web with Django, thank you very much. (APPLAUSE).

RIVO LAKS: Thanks, I am Rivo, I come from an Estonian product development agency, we use lots of Django in our products and we try to change the world. Here I am here to talk about Django and the real-time web and how the two link together.

So Django is ten years old and it is certainly a stable and mature framework but when you think about it and the web has changed so much during the last ten years, in 2005 we didn't have small personal computers in our pockets, mobile web didn't exist, Gmail just came out. It looks like that, paradox 1.0 was also something new. So the question is, is Django still relevant with all the change requirements that today's web puts on it?

Or has the pony become the dinosaur?

No I don't think so. Because, Django is actually really modular and really flexible and you can use it to you can use the parts that help you and will show that you can just add a real-time functionality into your application really easily and keep your application up to date so that it hits today’s needs.

So, a small disclaimer real-time web isn't obviously something that must be used everywhere. My Blog for example doesn't really need real-time updates but, on the other hand it also has some really good news cases where it makes the UI more flexible and fluid and makes the user experience better which is what matters in the end.

So, probably the easiest way that is you just need to push some updates from the server side into the client side when some data changes. This is commonly known as the Pub Sub. It is really easy to do in the sense that there are many external services that provide this functionality for you and it is always good use insisting services or libraries where possible you don't really want to reinvent the wheel, but you want to focus on your application and use whatever building blocks are there.

So some of the services include pusher and pusher is what I am going the show today. They all work pretty much the same way so it doesn't really matter that much which one you want to pick.

You just basically go the pusher website, sign up for a free account and first thing you do is you need to add some code to your server and it is really easy. First you create the pusher instance, give it your ID and credentials. Next you can already start sending messages with the trigger function. The trigger function has 3 important parameters. First we have the channel which is kind of like chat channels in the sense that when you send a message to the channel then all the clients which are subscribed to the channel and listening to it will receive this message. The second in this case show message is the message type itself or the name. So if we are building a chat we could have a show message we could also have join and leave in the names. Then we have the actual data that we want to send in this case, the message itself and so, with less than 20 lines of code we already have some way of proactively sending updates into the browser whenever something happens.

The next obviously we have to actually receive those messages on the client side in the browser. This is almost as easy, again we create the pusher instance then we say that we are interested in a certain channel or the message is sent to that channel and then, event handler which reacts to messages of a certain type in, in this case the show message.

Here, in simple demo we just show the model with the receive message.

I want to show a live demo here, there is one, I will give you the link and you can try it afterwards.

You probably also want to have the other direction you want to send the data from the client or the browser to the server and the easiest way to do that is actually bios requests, so you don't need anything fancy there, when the client or the user types something and wants to show a message you do the usual post request to the Django and then Django can push this message forward to all the clients listening.

So, as you can see, it is really easy to add some real-time functionality into your application, talk about the 20 lines of code would probably take about 5 minutes to integrate it and the external services are good again in the sense that you don't have to have your own infrastructure, you don't have to build the building blocks. You can instead use existing libraries to build your application and focus on what is important for you.

But, sometimes it is not really enough what the external services provide. For example, you might want to have faster messages or the full control over the messages that you send. This is where web sockets come in. So web sockets sees a protocol standardised in 2011 supported on every modern browser. What web sockets is give you the connection between the browser and the server and it is very low overhead, it is very fast. It also supports both text and binary data. In fact we used this binary web sockets where we used to it to work with street lights. So it is versatile.

Let's also look at how web sockets can be used and integrated within Django, using asyncio library, we will also be using web sockets package which gives you again the building blocks or primitives that you can use to just focus on sending and receiving the messages instead of implementing the web sockets protocol yourself.

Because the web sockets should run on a separate port from your main application, we are also going the need a custom server process for that.

I am skipping some of the code. But the important part is this, define a handler that gets called whenever a client connects. Then you can use the web socket to send and receive messages.

Again, very simple. You can expand on that to basically create the loop that reads messages from client and then processes them in some way like printing them out to the standard output.

On the client code web socket has pretty good standard API so you don't really need any third party library, you can use ... JRS, but also use simple rapper library, I am using something called sawkit.

When the connection is made, it sends a message to the server and waits for incoming messages and reacts to them. Again, showing the alert.

So I hope I have managed to show that Django and real-time web can relate really easily., easy to use external services like pusher. Also simple with websockets. It is not difficult but often some third party service will also get you off quite well.

There is also demo up at Djangocon.thorgate.eu. There is code. We also have a stand at the front, please come and see me if you are interested in more. I will tweet the location of the slides thank you. (APPLAUSE).

NEW SPEAKER: Does anyone have any questions?

FROM THE FLOOR: Not really a question, more a remark, there is a new ... called web bush and ... (INAUDIBLE) we have made an implementation that is called web push and there is an implementation, ... so if you want to ... you can use it. Basically, when the clients connect to the server, ... you can post ... web socket.

RIVO LAKS: Okay thank you, that is really interesting to know.

FROM THE FLOOR: What sort of things would you recommend if you wanted to have a mobile native type app.

RIVO LAKS: I think pusher has made ... it depends on your needs, for something pusher might be enough or you can use websockets.

FROM THE FLOOR: How would you handle anything other than a client ... you could (INAUDIBLE) something like a mobile app, so is it just for websites and browsers? Or can it lead to anything?

RIVO LAKS: Use it between any two, in the project we are using it to communicate between server and not browser, but many devices which controls street lamps. So, street lamps, so yes, it is quite versatile.

FROM THE FLOOR: Try out ... dragon, which is a library that is made for Django, using websockets.

RIVO LAKS: I have heard of it, but not tried it yet.

NEW SPEAKER: Okay, let's thank the speaker again.

(APPLAUSE).

So today for our final talk of the final session we have Jamie Hannaford who is a software engineer with rack space, and will be talking about making us more inclusive {applause}.

JAMIE HANNAFORD: So our industry is defined by its ability to solve technical problems. We produce software, tests, documentation, sometimes of immense complexity as a result. But, let's not fool ourselves the most technical problem in tech is not technical it's social. People are more difficult to work with than computers and to day I want to talk to you about diversity but also about inclusiveness.

So the first diversity I want to talk about is diversity in the workplace, the challenges we face and some of the techniques and this we can do to mitigate and overcome them. The second thing I want to talk about is open source communities and how we can make them more inclusive for everybody. Thirdly I want to talk about physical spaces, conferences and meet ups and how we can minimise exclusion there. So, I think the first question we really need to ask ourselves is what do I mean by diversity?

For me, diversity is not just about hiring more women. Gender is only one part of the picture. We should aim bigger by incorporating a wider spectrum of people that covers race, ethnicity, sexuality and religion, age, disability, class, educational background and bodily appearance. These are all truly important aspects of what it is to have a diverse team. And to answer the question of why is diversity important the simple answer is it makes your teams better. Fresh ideas and new perspectives are introduced. And we begin to think differently and make better decisions. And as a result, we produce more innovative products which drive business up for the overall company.

In fact there have actually been studies that prove this. For example, the Kelup School of Management a few years ago commissioned a study where they got 200 people and placed them into groups of 3 and they gave them a simple detective problem. They were given a set of interviews and they had to determine the likely murder suspect. Now each person was given the interviews and they basically had to come up with their own ideas but they were in groups of 3 from the same social background so the groups were fairly homogenous. After 5 minutes a fourth member joined the team. 50 per cent of the time it was from somebody from the same social background so 4 people who effectively thought the same. The other 50 per cent of the time it was a stranger an outsider which was effectively composing a diverse team. The results were startling they found because the diverse teams guessed the correct murder suspect with far greater accuracy than the homogenous group. And this wasn't just because of the influence of new ideas and fresh perspectives, what they found was it actually triggered more careful information processing, so there is a real, real compulsion and reason there for diversity in terms of what it gives our teams.

So, we know what diversity is and we know why it's important, so what concrete tings can we do in our workplace in our team to incorporate diversity? Well, unfortunately saying we care about diversity isn't really enough. Action speaks louder than words and one of the ways we can actually solve this problem, take a first step, is to reform a hiring process.

The first way, the first important aspect of when we start to think about how we can reform our hiring process is to look at the way we right job postings. Now the job posting is the primary interface of your company. It is the lens that allow prospective candidates to glimpse your internal culture. So if we want to write a really good job posting that's inclusive and will encourage diversity, what things do we need to do? Well, we need to focus on the tone, we need to ensure it is friendly, accessible and completely assumption free, we should avoid all the irrelevant clichés like rock star, hero, ninja, we're all tired of them now and they don't contribute any knowledge to job posting itself. If you employer actually is willing to have a commitment to diversity make it explicit, there is nothing wrong with being open about it because it encourages people to realise what type of company you have. And lastly you really need to focus on offering things that people would genuinely want. So, not everybody cares about having free red bull and coke in the fridge or a football table. Things that people might want is flexible hours or remote working, childcare assistance, accessible and wheelchair friendly facilities, mental healthcare coverage, trans inclusive healthcare coverage, generous parental leave. These are all truly important things people look for in a modern diverse company.

The second aspect of how we can reform hiring is to look at the way we interview people. Now the technical interview has traditionally been a major arena where implicit unconscious bias enters the picture because we tend to favour people from the same background who went to university, studied computer science and as a result we kind of enforce homogeneity. We don't promote diverse thinkers because we have a very monolithic way of approaching this.

Now, when we ask people to code on a white board or prove to us that theoretical knowledge or algorhythmic integrity we really have to ask ourselves whether me want to be part of an engineering team that was primarily chosen by its ability to write code on a white board. For me it's kind of irrelevant. That's not really what I do on a day to day basis as a programmer, I don't really know anyone else who does that. So outspoken confidence and having an in depth and theoretical knowledge of algorithms is not the most important skills a team needs.

Also think that we shouldn't be afraid of trying to redress the balance a bit because when we start a conversation or a discussion about changing hiring practices it can be really difficult with your team because it actually has a tendency to reveal potential internal problems with your own culture, culture and the teams culture itself.

So, what are the things we do to make these things better to make them more inclusive? Well, let's be conversational, let's not be adversarial. Nobody should be set up for failure. Instead we need interviewers that are more self-reflective and less alpha male, we need more effective sensitive inclusive interviews. Also let's give the candidate something that will accurately measure their performance on a day to day basis so for example pair them, give them a problem and run through a - let them run through how they might solve the problem, can they debug? What is their code review process like? Can they fix an elusive issue? can they optimise a performance bottleneck? How will they re-influence something if they have a chance? These are all relevant questions that will accurately measure their performance of your team not ask them to scrawl stuff on a white board and actually Google they found zero correlation between technical interview performance and job performance so it just goes to show how meaningless the white board interview truly is.

So, if we move beyond hire itself and start to think about some of the challenges that are still left in our teams, one of them is a culture of fear. What I mean by that is the fear of asking questions. The fear of thinking you’re going to look stupid if you do, fear of thinking you don't know enough, it's a real fear of pretty much everyone in the technical industry and the fact is our industry is evolving at an extremely fast pace, we often expect ourselves to keep up with this constant onslaught of new technologies and knowledge but there is a constant threat of being left behind and out of our depth. We all face a knowledge deficit and people who start to {inaudible} are beginners, they often get frustrated really easily because they see other developers using really could technologies and they question themselves and say why bother? What can I do? I can do nothing. But guess what, we've all been there, we've all been beginners and we've all had that self-doubt. But it's OK, it's OK to be a beginner and it's OK to have these feelings of inadequacy. Learning to programme is difficult but so is anything that's worthwhile.

So, apart from beginners we also have a problem of fear with seasoned developers, experienced developers. One of the very common things we're starting to talk about in our industry is the idea of imposter syndrome which if you are not aware is the psychological phenomenon which arises from an incorrect assessment of one’s own abilities so for example if you are an 8 or a 9 out of 10 programmer you yourself only think you are a 3 so it's an inability to really understand your own merits basically. And how does it manifest itself? If you are on a team with somebody who has imposter syndrome they often won't share knowledge because they don't think they have anything worthwhile to contribute, they don't speak at conferences, they don't submit talk proposals, they don't write blog requests, they don't collaborate contribute to open source and they don't apply for jobs. But beginner’s fixer these problem is more than personal responsibility, we can't ask people to solve it by themselves. I think it's endemic of a wider communal problem that we all have a responsibility to help with, we should all help each other.

So, Sasha Mundy gave a great talk at Python this year about how to give and get technical help and came up with a great line where she said: if people are afraid to ask for help it prevents from them stretching, if you don't stretch you don't grow. If people don't grow your team stagnates and your company doesn't build amazing things.

I think that perfectly summarises the struggles we all face when asking questions saw for people with the little voice inside their head telling them they're not good enough we want you to fight it, we need to have the braveness to overcome it and having that struggle of trying to overcome it too and for mentors we really should be aware of it, so if we're on a theme and we're mentoring people with no self-confidence or people who are constantly questioning themselves be aware of it, reward question asking and realise how brave it is and if they are giving and asking too many questions shape their behaviour, teach them how to fish instead of just giving them fish.

So to switch focus now to look at open source communities, the most relevant and pertinent question me need to ask is why don't more people contribute to open source whether it's Django or PHP or ruby? For me the answer is that not everybody is aware of the full range of contributing options so for example you can contribute documentation, you can write tests, you can write blog posts, you can speak about something, you can design an interface or just give feedback. These are all equally valid ways of contributing to open source but people tend to get hung up on the code slinging code they tend to think it's the only way to contribute and it's not.

Another vastly underrated way of contributing to the community is organising conferences and events because it often takes months of very careful hard work to set up and it's as much of a solid contribution as writing a new feature for Django.

Another reason more people don't contribute to open source is often people don't have the luxury of time and opportunity. Women for example are far more likely to be a primary care giver not only for children but also to ageing or ailing relatives. On childcare alone for example they spend more than twice as much time per day as fathers do and there are other reasons, some people might have medical conditions. Some people might get paid very low cannot sacrifice any contribution to do external things, some people have long commute and when they get home are so exhausted they can't contribute to open source. So I was reading a blog post a few months ago on modern new culture and the question asked was why don't you contribute to open source? And Anna who wrote the book post she said, well, I tried and people were unwelcoming and even cruel.

So this is us line from Julie Pegano also at Python where she talks about imposter syndrome and she was referring to very bad habit we have in our industry of elevating certain people to God like status, we think they're inscrutable and their code is flaw less we think they make incontrovertible technical decisions and think their status is untouchable but the direct consequence is that we doubt ourselves negate our own contributions and shy away from getting involved so her message was skill kill the heroes not necessarily that because she would violate the code of conduct but she said we need to leave the person behind and see them as a realistic person and it helps us equalise our own contributions and helps rationalise our own feelings of ourselves. Another really sort of insidious thing that happens in our industry is meritocracy the idea that power is bestowed by technical contributions but for me that's deeply flawed because it tends to rule one type of contribution. I said about multiple different ways to contribute. Meritocracy recognises code and that's it and another flaw is it assumes everyone is on the same level and has the same level of access to opportunity, time and money and we all know that's not the same, people aren't equal. So, what can we do to enable people and give them confidence in open source?

For me the biggest tip I can say is write a good contributing guide that formalises the transparent process that people need to abide by to get a patch merged; if you write it down and make it explicit there is no knowledge deficit there, people on the same level and know exactly what to do to get involved. We also need empathy and patience with issues and bug reports because if someone has taken the time to report a problem they have done that because they want to help the eco system and want to help strengthen the product itself and we should have honest intact, we should be appreciative of the fact that somebody has taken the time to write a feature.

Another thing I do with all of my projects is I deliver certain feedback really pedantic feedback through computers so if I have a very strict policy through syntax for example so in Python with indentation that kind of pedantic feedback is a lot better coming from an automated system or computer because if somebody thinks you are rejecting their contribution because they're missing a comma, they're probably going to walk away and say I don't want any part of this ridiculous charade but if it comes from a computer because they're stupid it's OK like I will accept that feedback and change it and re-push and everything is gravy but if it comes from a person then I will probably doubt that a little bit more.

Another thing which I think is really important is to stop denigrating peoples doling interests. I am a PHP developer and every time I introduce myself as a PHP developer I can see the eyes roll back there is an implicit judgment on me and I don't understand it I don't understand how we can judge somebody's choice of tool aunt language based on our own assumptions, I think it's wrong, we need to be a lot more open. Another example is word press developers, they get a lot of flak because they have very strong opinions about the way the code is structured but the way the code is structured is completely irrelevant, they're using that tool for a specific purpose and if it's good for them it's could enough for me and it should be good enough for you too.

And the final thing to really, that is essential to help improve inclusiveness is to document well. We had this idea earlier of documentation driven development and I completely agree, documentation is at the heart of all projects so we should make potential and assumption free knowledge sharing keep component of what we value nothing is ever obvious or easy and for extra bonus points you can do things like provide non English translations, you can focus on accessibility for visual impaired users, these are all really cool things you can do to make your project as inclusive as possible for as wide spectrum of people.

Lastly to focus on physical events themselves, what can we do to tackle exclusion? Now, I think it's very important to reason Django is pretty much the shining light in our industry of how to do it right, they have a code of conduct which basically helps define and make transparent what kind of behaviour is deemed exclusionary or threatening so everybody is on the same level and can no longer plead ignorance. Codes of conduct help formalise the support system for marginalised people and allow them to contact organisers directly and also one of the things I regularly hear about the reason not to have a code of conduct is people say nobody else before has reported harassment but just because that's happened doesn't mean harassment doesn't happen, it could mean somebody didn't feel confident enough to report a bad thing.

So to wrap up, I think we should remember the 4 points I talked about today. We need to change the way that we hire, we need to remove the culture of fear, we should formalise the transparent process of contributing to open source projects and we should make an attempt supported events that value diversity.

We all have the ability to change ourselves and by association we have the ability to change the communities we all belong to. Bell hooks once said dominating culture has tried to keep us all afraid to make us choose safely instead of risk, same risk instead of diversity, moving through that fear finding out what connects it, revelling in our differences this the world that brings us closer gives us a world of shared values a meaningful community. I will leave you with that quote thank you. {applause}.

DANIELE PROCIDA: Thank you very much I think that was the perfect way to end today so thanks.

Just before we all leave the room a couple of very brief announcements.

Actually I'm so sorry, Jamie, sorry, I didn't ask - there must be some questions for Jamie. I'm sorry I didn't mean to - I'm sure there must be questions for Jamie come and stand here so - yes please.

NEW SPEAKER: Hi Jamie thanks so much for that great talk. I get from the sense of your talk that you are deeply into diversity because you think it is the right thing to do but I was interested in how you kind of led the beginning of your talk with the idea that - I mean I kind of promulgated it in my talk too like when you have diverse teams we make better things and so I'm wondering do you ever kind of use that line to kind of finesse diversity to some people instead of - because I come from an ethics background I believe diversity is the right thing to do. Can you talk about the way you play with that sometimes?

JAMIE HANNAFORD: So, I think that it is very important when you believe so strongly in something and you are trying to make changes to be sensitive to the fact that not everybody is going to have the same ethical framework as you. They are going to think differently because they are going to have different moral and political beliefs so I think it is important not to hammer my framework on somebody. I find the presenting facts and psychological study is kind of a neat way out because our industry loves numbers, they love stats, they love very technical things that we can't argue with, if you give them that, how can they say no? It is a better way of trying to achieve change than just trying to talk about ethics but I think both are important.

NEW SPEAKER: Truly tremendous talk, really well done. It was really refreshing and I think it is a real testament to Daniel's vision that we have got so much inclusivity at this meeting.

(APPLAUSE).

I am a research astronomer and we, there is a group of us who crossed over into this sort of dark world of software writing and the some of the lessons that we brought back are including things like codes of conduct in our meetings and it is really, really important to us that you guys are leading the way in this. Because we can benefit from this. The epidemic community for a long time tried to a lot of the hinges, like you said, nobody has said they have been harassed, why? All those sorts of things so, that was really what I wanted to say that it actually it is wider than just this community. It does trickle back into other areas and it is really, really important certainly to us.

JAMIE HANNAFORD: Some of the problems I talked about some doubts and voices, they are applicable in every industry, I think it is great that our tech industry can be seen like an archetype of how others can act, we all have a responsibility because we are on the front lines, we are the ones defining future generations of other industries too. We need to pitch in and make it better.

DANIELE PROCIDA: Last one.

NEW SPEAKER: Thank you for your talk, so my question is, if you have managed an open-source project, ... a guideline not just for the coding and the patches for doing research, ... and all sorts of contributions.

JAMIE HANNAFORD: Are you talking about personally? Do I -- have I seen others.

FROM THE FLOOR: Have you seen?

JAMIE HANNAFORD: Not that I can recall, I can look into it. The types of projects I maintain I make it key priority when I start them to focus on unfortunately just on code contributions but now that you have raised this issue of different types of contributions I think that is something I will start thinking about with my projects too. I can be on the lookout if I see ones that are of particular interest.

DANIELE PROCIDA: Right thank you once again.

(APPLAUSE).

Okay, it is the end of our first day of our open day. So I have got a lot of thank you’d, firstly thank you all for coming especially if you weren't part of the conference and came to visit our open day, we are really glad you are able to be our guest here today, maybe we will see you at some other events.

Thank you to our speakers and the workshop leaders today, who have, I have seen most of the talks, not seen all of them. The workshops were all really busy, so thank you for everybody who put so much effort into that too.

Our volunteers, are any of the volunteers in the audience? No, well you will have to thank them, look, yes there are, Damien has been yes, you Damien, you will have to represent the volunteers they have done a fantastic job and grateful for everything they have done.

The people who provided technical services, our speech to text reporters, people doing the filming, people looking after the AV for us and people looking after the network for us, so thank you very helping keeping today running.

Once more, thank you to Cardiff University if you can send a little thank you to Cardiff University that would be nice, they have put an awful lot into this event. Who is a sponsor here? Representing a sponsor, come on I know you are, thank you very much because you are also one of the, you also make this possible it would not be possible or not possible like this were it not for the involvement and active participation of sponsors, thanks very much to all of those. (APPLAUSE).

I hope you have enjoyed today, I hope that you will be able to come to one or the other of the meals that we have got on tonight. You do need tickets for those, tickets for the VFS are still available from the tickets page on the website. Whichever of those venues, be there by 7:15 this evening, if you are going the the VFS, they don't serve alcohols but you are welcome to take in your own bottles. They don't serve alcohol at the clink for entirely different reasons! see you later or tomorrow!