tutorial.djangogirls.org

How the Internet works · Django Girls Tutorial

DjangoGirls

4-5 minuti

> For readers at home: this chapter is covered in the How the Internet Works video.
>
> This chapter is inspired by the talk "How the Internet works" by Jessica McKellar (http://web.mit.edu/jesstess/www/).

We bet you use the Internet every day. But do you actually know what happens when you type an address like https://djangogirls.org into your browser and press `enter`?

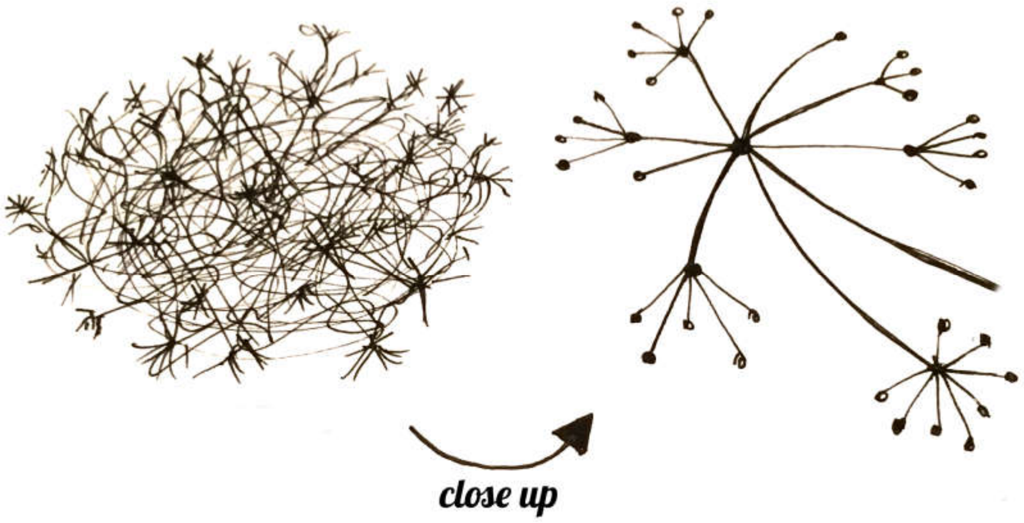
The first thing you need to understand is that a website is just a bunch of files saved on a hard disk. Just like your movies, music, or pictures. However, there is one part that is unique for websites: they include computer code called HTML.

If you're not familiar with programming it can be hard to grasp HTML at first, but your web browsers (like Chrome, Safari, Firefox, etc.) love it. Web browsers are designed to understand this code, follow its instructions, and present these files that your website is made of, exactly the way you want.

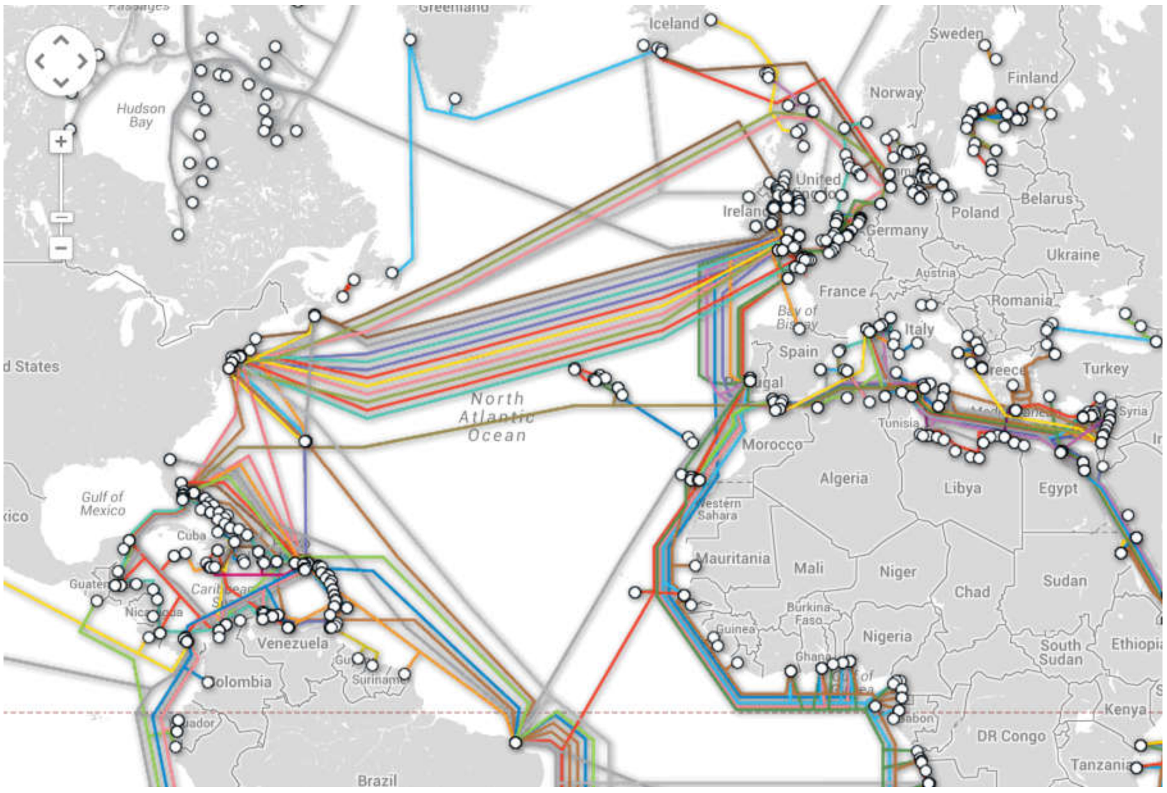
As with every file, we need to store HTML files somewhere on a hard disk. For the Internet, we use special, powerful computers called *servers*. They don't have a screen, mouse or a keyboard, because their main purpose is to store data and serve it. That's why they're called *servers* – because they *serve* you data.

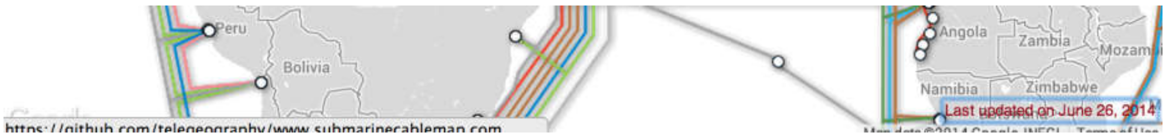
OK, but you want to know how the Internet looks, right?

We drew you a picture! It looks like this:



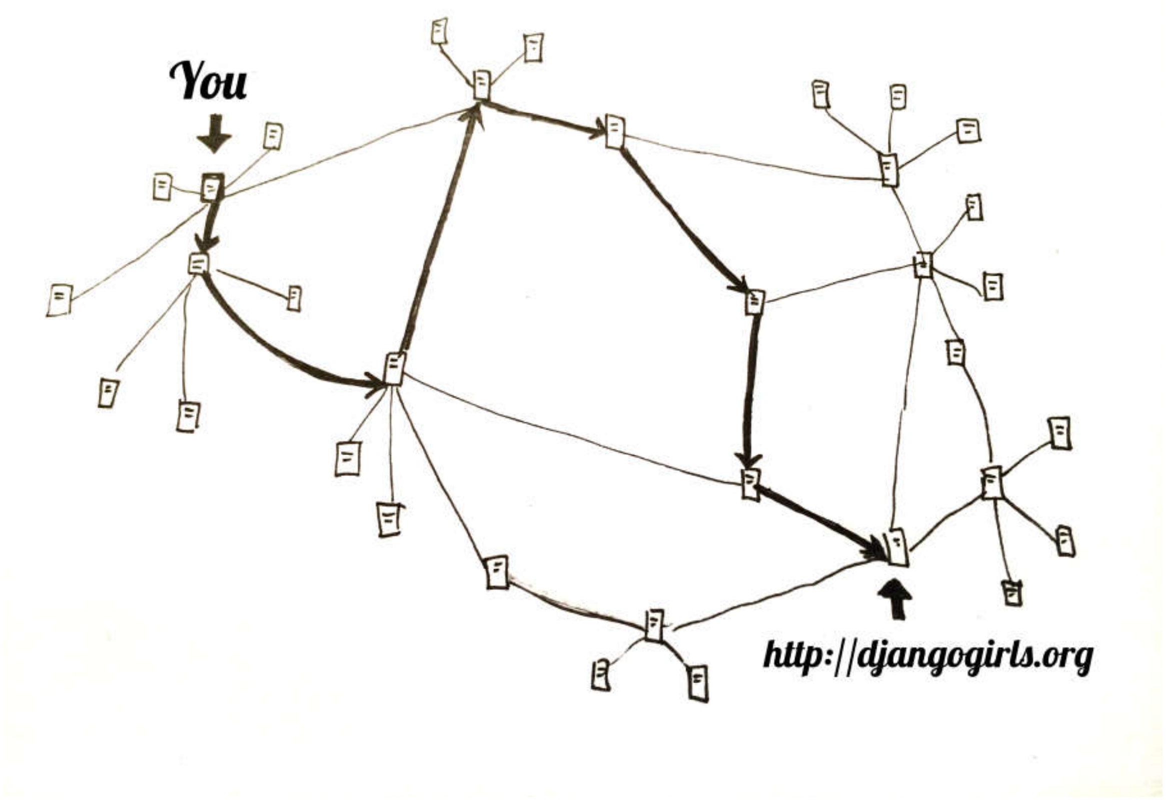
Looks like a mess, right? In fact it is a network of connected machines (the above-mentioned *servers*). Hundreds of thousands of machines! Many, many kilometers of cables around the world! You can visit a Submarine Cable Map website (http://submarinecablemap.com) to see how complicated the net is. Here is a screenshot from the website:




Peru
Bolivia
Angola
Zambia
Mozam
Namibia
Zimbabwe
Last updated on June 26, 2014
https://github.com/telegeography/www.submarinecablemap.com

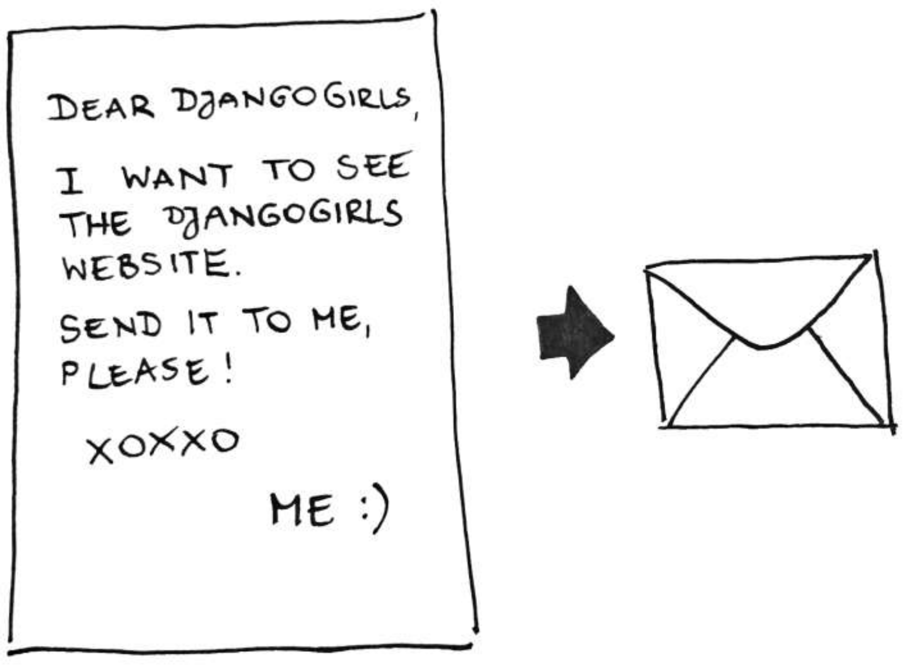
It is fascinating, isn't it? But obviously, it is not possible to have a wire between every machine connected to the Internet. So, to reach a machine (for example, the one where https://djangogirls.org is saved) we need to pass a request through many, many different machines.

It looks like this:



Imagine that when you type https://djangogirls.org, you send a letter that says: "Dear Django Girls, I want to see the djangogirls.org website. Send it to me, please!"

Your letter goes to the post office closest to you. Then it goes to another that is a bit nearer to your addressee, then to another, and another until it is delivered at its destination. The only unique thing is that if you send many letters (*data packets*) to the same place, they could go through totally different post offices (*routers*). This depends on how they are distributed at each office.



Yes, it is as simple as that. You send messages and you expect some response. Of course, instead of paper and pen you use bytes of data, but the idea is the same!

Instead of addresses with a street name, city, zip code and country name, we use IP addresses. Your computer first asks the DNS (Domain Name System) to translate djangogirls.org into an IP address. It works a little bit like old-fashioned phonebooks where you can look up the name of the person you want to contact and find their phone number and address.

When you send a letter, it needs to have certain features to be delivered correctly: an address, a stamp, etc. You also use a language that the receiver understands, right? The same applies to the *data packets* you send to see a website. We use a protocol called HTTP (Hypertext Transfer Protocol).

So, basically, when you have a website, you need to have a *server* (machine) where it lives. When the *server* receives an incoming *request* (in a letter), it sends back your website (in another letter).

Since this is a Django tutorial, you might ask what Django does. When you send a response, you don't always want to send the same thing to everybody. It is so much better if your letters are personalized, especially for the person that has just written to you, right? Django helps you with creating these personalized, interesting letters. :)

Enough talk – time to create!

tutorial.djangogirls.org

Installation · Django Girls Tutorial

DjangoGirls

17-21 minuti

If you're doing the tutorial at home, not at one of the Django Girls events, you can completely skip this chapter now and go straight to the How the Internet works chapter.

This is because we cover these things in the whole tutorial anyway, and this is just an additional page that gathers all of the installation instructions in one place. The Django Girls event includes one "Installation evening" where we install everything so we don't need to bother with it during the workshop, so this is useful for us.

If you find it useful, you can follow through this chapter too. But if you want to start learning things before installing a bunch of stuff on your computer, skip this chapter and we will explain the installation part to you later on.

Good luck!

In the workshop you will be building a blog, and there are a few setup tasks in the tutorial which would be good to work through beforehand so that you are ready to start coding on the day.

Chromebook setup (if you're using one)

You can skip right over this section if you're not using a Chromebook. If you are, your installation experience will be a little different. You can ignore the rest of the installation instructions.

Cloud 9

Cloud 9 is a tool that gives you a code editor and access to a computer running on the Internet where you can install, write, and run software. For the duration of the tutorial, Cloud 9 will act as your *local machine*. You'll still be running commands in a terminal interface just like your classmates on OS X, Ubuntu, or Windows, but your terminal will be connected to a computer running somewhere else that Cloud 9 sets up for you.

1. Install Cloud 9 from the Chrome web store
2. Go to c9.io
3. Sign up for an account
4. Click *Create a New Workspace*
5. Name it *django-girls*
6. Select the *Blank* (second from the right on the bottom row with orange logo)

   Now you should see an interface with a sidebar, a big main window with some text, and a small window at the bottom that looks something like this:

   Cloud 9

   ```
   yourusername:~/workspace $
   ```

   This bottom area is your *terminal*, where you will give the computer Cloud 9 has prepared for you instructions. You can resize that window to make it a bit bigger.

Virtual Environment

A virtual environment (also called a virtualenv) is like a private box we can stuff useful computer code into for a project we're working on. We use them to keep the various bits of code we want for our various projects separate so things don't get mixed up between projects.

In your terminal at the bottom of the Cloud 9 interface, run the following:

Cloud 9

```
sudo apt update
sudo apt install python3.6-venv
```

If this still doesn't work, ask your coach for some help.

Next, run:

Cloud 9

```
mkdir djangogirls
cd djangogirls
python3.6 -mvenv myvenv
source myvenv/bin/activate
pip install django~=1.11.0
```

(note that on the last line we use a tilde followed by an equal sign: ~=).

Github

Make a Github account.

PythonAnywhere

The Django Girls tutorial includes a section on what is called Deployment, which is the process of taking the code that powers your new web application and moving it to a publicly accessible computer (called a server) so other people can see your work.

This part is a little odd when doing the tutorial on a Chromebook since we're already using a computer that is on the Internet (as opposed to, say, a laptop). However, it's still useful, as we can think of our Cloud 9 workspace as a place or our "in progress" work and Python Anywhere as a place to show off our stuff as it becomes more complete.

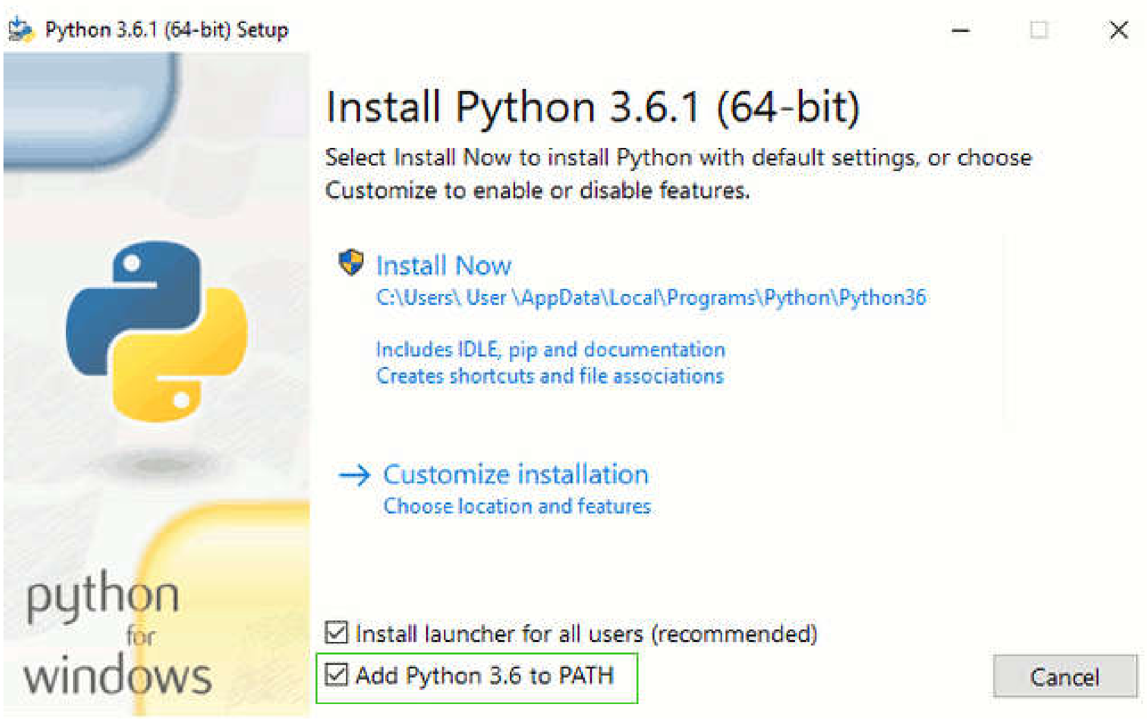
Thus, sign up for a new Python Anywhere account at www.pythonanywhere.com.

> For readers at home: this chapter is covered in the Installing Python & Code Editor video.
>
> This section is based on a tutorial by Geek Girls Carrots (https://github.com/ggcarrots/django-carrots)

Django is written in Python. We need Python to do anything in Django. Let's start by installing it! We want you to install Python 3.6, so if you have any earlier version, you will need to upgrade it.

First check whether your computer is running a 32-bit version or a 64-bit version of Windows, by pressing the Windows key + Pause/Break key which will open your System info, and look at the "System type" line. You can download Python for Windows from the website https://www.python.org/downloads/windows/. Click on the "Latest Python 3 Release - Python x.x.x" link. If your computer is running a **64-bit** version of Windows, download the **Windows x86-64 executable installer**. Otherwise, download the **Windows x86 executable installer**. After downloading the installer, you should run it (double-click on it) and follow the instructions there.

One thing to watch out for: During the installation you will notice a window marked "Setup". Make sure you tick the "Add Python 3.6 to PATH" checkbox and click on "Install Now", as shown here:



Install Python 3.6.1 (64-bit)

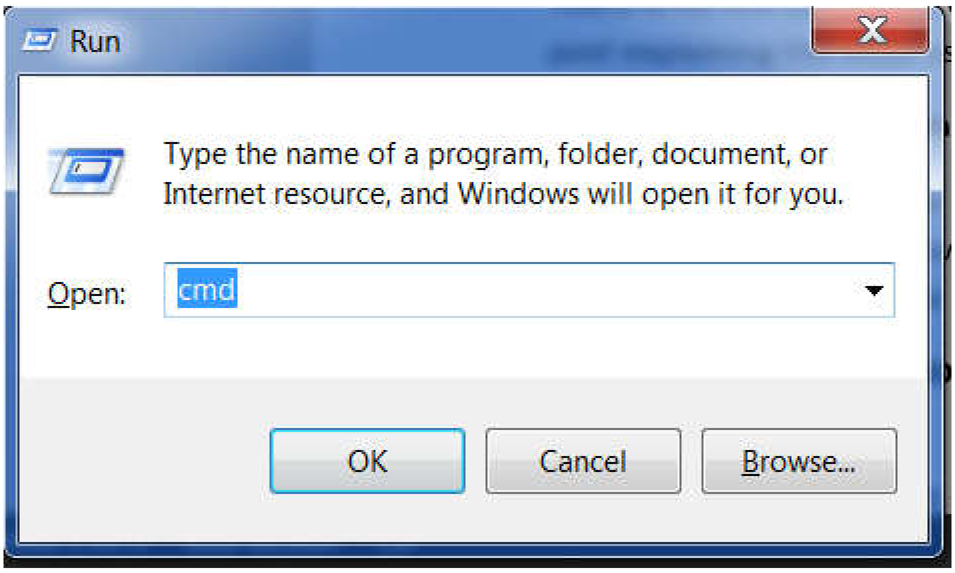
Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

C:\Users\ User \AppData\Local\Programs\Python\Python36

Includes IDLE, pip and documentation
Creates shortcuts and file associations

☑ Install launcher for all users (recommended)
☑ Add Python 3.6 to PATH

In upcoming steps, you'll be using the Windows Command Line (which we'll also tell you about). For now, if you need to type in some commands, go to Start menu → Windows System → Command Prompt. You can also hold in the Windows key and press the "R"-key until the "Run" window pops up. To open the Command Line, type "cmd" and press enter in the "Run" window. (On newer versions of Windows, you might have to search for "Command Prompt" since it's sometimes hidden.)



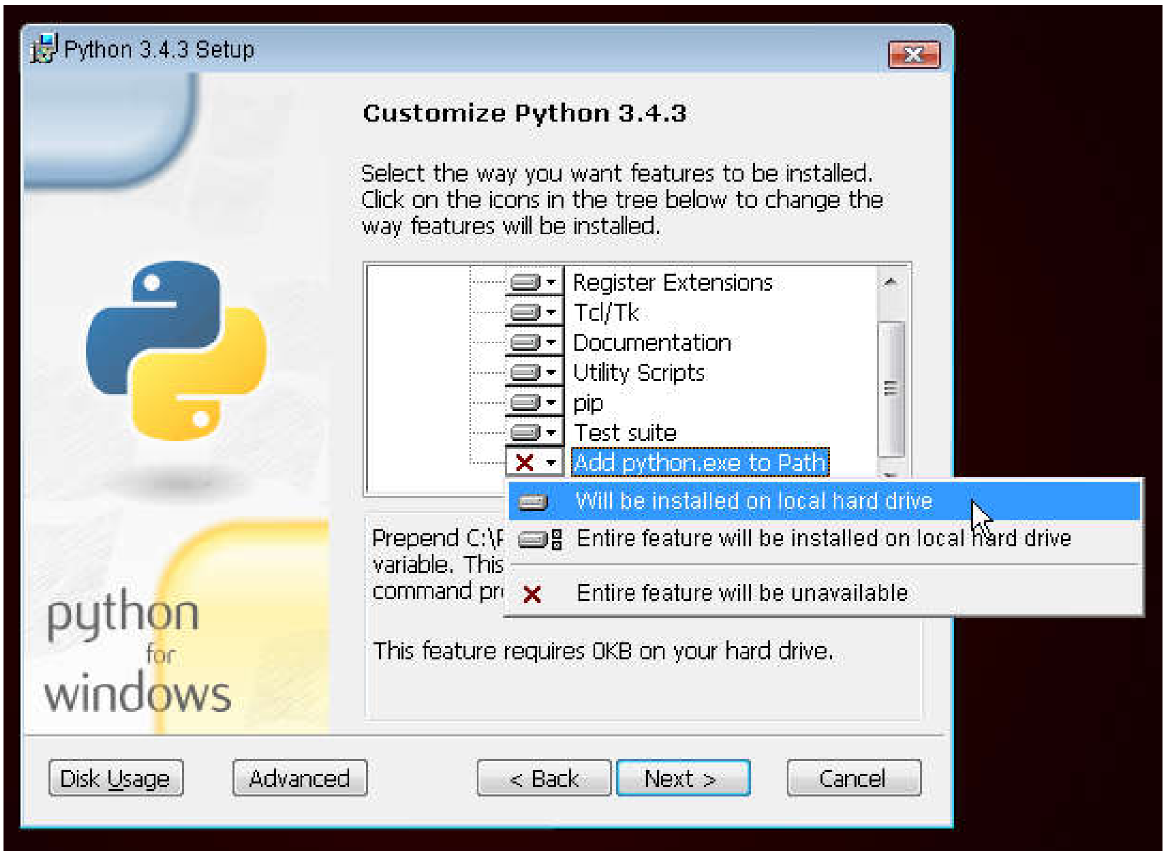
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: cmd

Note: if you are using an older version of Windows (7, Vista, or any older version) and the Python 3.6.x installer fails with an error, you can try either:

1. install all Windows Updates and try to install Python 3.6 again; or
2. install an older version of Python, e.g., 3.4.6.

If you install an older version of Python, the installation screen may look a bit different than shown above. Make sure you scroll down to see "Add python.exe to Path", then click the button on the left and pick "Will be installed on local hard drive":


Python 3.4.3 Setup
Customize Python 3.4.3
Select the way you want features to be installed. Click on the icons in the tree below to change the way features will be installed.
Register Extensions
Tcl/Tk
Documentation
Utility Scripts
pip
Test suite
Add python.exe to Path
Will be installed on local hard drive
Entire feature will be installed on local hard drive
Entire feature will be unavailable
Prepend C:\F
variable. This
command pr
This feature requires 0KB on your hard drive.
python for windows
Disk Usage
Advanced
< Back
Next >
Cancel

Note Before you install Python on OS X, you should ensure your Mac settings allow installing packages that aren't from the App Store. Go to System Preferences (it's in the Applications folder), click "Security & Privacy," and then the "General" tab. If your "Allow apps downloaded from:" is set to "Mac App Store," change it to "Mac App Store and identified developers."

You need to go to the website https://www.python.org/downloads/release/python-361/ and download the Python installer:

- Download the *Mac OS X 64-bit/32-bit installer* file,

- Double click *python-3.6.1-macosx10.6.pkg* to run the installer.

  It is very likely that you already have Python installed out of the box. To check if you have it installed (and which version it is), open a console and type the following command:

  command-line

  ```
  $ python3 --version
  Python 3.6.1
  ```

  If you have a different 'micro version' of Python installed, e.g. 3.6.0, then you don't have to upgrade. If you don't have Python installed, or if you want a different version, you can install it as follows:

  Install Python: Debian or Ubuntu

  Type this command into your console:

  command-line

  ```
  $ sudo apt-get install python3.6
  ```

  Use this command in your console:

  command-line

  ```
  $ sudo dnf install python3
  ```

  If you're on older Fedora versions you might get an error that the command dnf is not found. In that case you need to use yum instead.

  Use this command in your console:

  command-line

  ```
  $ sudo zypper install python3
  ```

  Verify the installation was successful by opening a command prompt and running the `python3` command:

command-line

```
$ python3 --version
Python 3.6.1
```

NOTE: If you're on Windows and you get an error message that `python3` wasn't found, try using `python` (without the `3`) and check if it still might be a version of Python 3.6.

If you have any doubts, or if something went wrong and you have no idea what to do next, please ask your coach! Sometimes things don't go smoothly and it's better to ask for help from someone with more experience.

> Part of this section is based on tutorials by Geek Girls Carrots (https://github.com/ggcarrots/django-carrots).
>
> Part of this section is based on the django-marcador tutorial licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. The django-marcador tutorial is copyrighted by Markus Zapke-Gründemann et al.

Virtual environment

Before we install Django we will get you to install an extremely useful tool to help keep your coding environment tidy on your computer. It's possible to skip this step, but it's highly recommended. Starting with the best possible setup will save you a lot of trouble in the future!

So, let's create a **virtual environment** (also called a *virtualenv*). Virtualenv will isolate your Python/Django setup on a per-project basis. This means that any changes you make to one website won't affect any others you're also developing. Neat, right?

All you need to do is find a directory in which you want to create the `virtualenv`; your home directory, for example. On Windows it might look like `C:\Users\Name\` (where `Name` is the name of your login).

> **NOTE:** On Windows, make sure that this directory does not contain accented or special characters; if your username contains accented characters, use a different directory, for example `C:\djangogirls`.

For this tutorial we will be using a new directory `djangogirls` from your home directory:

command-line

```
$ mkdir djangogirls
$ cd djangogirls
```

We will make a virtualenv called `myvenv`. The general command will be in the format:

command-line

```
$ python3 -m venv myvenv
```

Virtual environment: Windows

To create a new `virtualenv`, you need to open the command prompt and run `python -m venv myvenv`. It will look like this:

command-line

```
C:\Users\Name\djangogirls> python -m venv myvenv
```

Where `myvenv` is the name of your `virtualenv`. You can use any other name, but stick to lowercase and use no spaces, accents or special characters. It is also good idea to keep the name short – you'll be referencing it a lot!

Virtual environment: Linux and OS X

Creating a `virtualenv` on both Linux and OS X is as simple as running `python3 -m venv myvenv`. It will look like this:

command-line

```
$ python3 -m venv myvenv
```

`myvenv` is the name of your `virtualenv`. You can use any other name, but stick to lowercase and use no spaces. It is also good idea to keep the name short as you'll be referencing it a lot!

> **NOTE:** On some versions of Debian/Ubuntu you may receive the following error:
>
> command-line
>
> ```
> The virtual environment was not created
> successfully because ensurepip is not available.
> On Debian/Ubuntu systems, you need to install
> the python3-venv package using the following
> command.
>    apt-get install python3-venv
> You may need to use sudo with that command.
> After installing the python3-venv package,
> recreate your virtual environment.
> ```
>
> In this case, follow the instructions above and install the `python3-venv` package:
>
> command-line
>
> ```
> $ sudo apt-get install python3-venv
> ```
>
> **NOTE:** On some versions of Debian/Ubuntu initiating the virtual environment like this currently gives the following error:
>
> command-line
>
> ```
> Error: Command '['/home/eddie/Slask/tmp/venv
> /bin/python3', '-Im', 'ensurepip', '--upgrade',
> '--default-pip']' returned non-zero exit status
> ```

> ```
> 1
> ```
>
> To get around this, use the `virtualenv` command instead.
>
> command-line
>
> ```
> $ sudo apt-get install python-virtualenv
> $ virtualenv --python=python3.6 myvenv
> ```
>
> **NOTE:** If you get an error like
>
> command-line
>
> ```
> E: Unable to locate package python3-venv
> ```
>
> then instead run:
>
> command-line
>
> ```
> sudo apt install python3.6-venv
> ```

Working with virtualenv

The command above will create a directory called `myvenv` (or whatever name you chose) that contains our virtual environment (basically a bunch of directory and files).

Working with virtualenv: Windows

Start your virtual environment by running:

command-line

```
C:\Users\Name\djangogirls> myvenv\Scripts
\activate
```

> **NOTE:** on Windows 10 you might get an error in the Windows PowerShell that says `execution of scripts is disabled on this system`. In this case, open another Windows PowerShell with the "Run as Administrator" option. Then try typing the following command before starting your virtual environment:

> command-line
>
> ```
> C:\WINDOWS\system32> Set-ExecutionPolicy
> -ExecutionPolicy RemoteSigned
>     Execution Policy Change
>     The execution policy helps protect you from
> scripts that you do not trust. Changing the
> execution policy might expose you to the
> security risks described in the
> about_Execution_Policies help topic at
> http://go.microsoft.com/fwlink/?LinkID=135170.
> Do you want to change the execution policy? [Y]
> Yes  [A] Yes to All  [N] No  [L] No to All  [S]
> Suspend  [?] Help (default is "N"): A
> ```

Working with virtualenv: Linux and OS X

Start your virtual environment by running:

command-line

```
$ source myvenv/bin/activate
```

Remember to replace `myvenv` with your chosen `virtualenv` name!

> **NOTE:** sometimes `source` might not be available. In those cases try doing this instead:
>
> command-line
>
> ```
> $ . myvenv/bin/activate
> ```

You will know that you have `virtualenv` started when you see that the prompt in your console is prefixed with `(myvenv)`.

When working within a virtual environment, `python` will automatically refer to the correct version so you can use `python` instead of `python3`.

OK, we have all important dependencies in place. We can finally install Django!

Installing Django

Now that you have your `virtualenv` started, you can install Django.

Before we do that, we should make sure we have the latest version of `pip`, the software that we use to install Django:

command-line

```
(myvenv) ~$ pip install --upgrade pip
```

Then run `pip install django~=1.11.0` (note that we use a tilde followed by an equal sign: `~=`) to install Django.

command-line

```
(myvenv) ~$ pip install django~=1.11.0
Collecting django~=1.11.0
  Downloading Django-1.11.3-py2.py3-none-any.whl
(6.8MB)
Installing collected packages: django
Successfully installed django-1.11.3
```

Installing Django: Windows

If you get an error when calling pip on Windows platform, please check if your project pathname contains spaces, accents or special characters (for example, `C:\Users\User Name\djangogirls`). If it does, please consider using another place without spaces, accents or special characters (suggestion: `C:\djangogirls`). Create a new virtualenv in the new directory, then delete the old one and try the above command again. (Moving the virtualenv directory won't work since virtualenv uses absolute paths.)

Installing Django: Windows 8 and Windows 10

> Your command line might freeze after when you try to install Django. If this happens, instead of the above command use:
>
> command-line
>
> ```
> C:\Users\Name\djangogirls> python -m pip install
> django~=1.11.0
> ```

> If you get an error when calling pip on Ubuntu 12.04 please run `python -m pip install -U --force-reinstall pip` to fix the pip installation in the virtualenv.

That's it! You're now (finally) ready to create a Django application!

There are a lot of different editors and it largely boils down to personal preference. Most Python programmers use complex but extremely powerful IDEs (Integrated Development Environments), such as PyCharm. As a beginner, however, that's probably less suitable; our recommendations are equally powerful, but a lot simpler.

Our suggestions are below, but feel free to ask your coach what their preferences are – it'll be easier to get help from them.

Gedit

Gedit is an open-source, free editor, available for all operating systems.

Download it here

Sublime Text 3

Sublime Text is a very popular editor with a free evaluation period. It's easy to install and use, and it's available for all operating systems.

Download it here

Atom

Atom is an extremely new code editor created by GitHub. It's free, open-source, easy to install and easy to use. It's available for Windows, OS X and Linux.

Download it here

Why are we installing a code editor?

You might be wondering why we are installing this special code editor software, rather than using something like Word or Notepad.

The first reason is that code needs to be **plain text**, and the problem with programs like Word and Textedit is that they don't actually produce plain text, they produce rich text (with fonts and formatting), using custom formats like RTF (Rich Text Format).

The second reason is that code editors are specialized for editing code, so they can provide helpful features like highlighting code with color according to its meaning, or automatically closing quotes for you.

We'll see all this in action later. Soon, you'll come to think of your trusty old code editor as one of your favorite tools. :)

Git is a "version control system" used by a lot of programmers. This software can track changes to files over time so that you can recall specific versions later. A bit like the "track changes" feature in Microsoft Word, but much more powerful.

Installing Git

You can download Git from git-scm.com. You can hit "next" on all steps except for one; in the fifth step entitled "Adjusting your PATH environment", choose "Use Git and optional Unix tools from the Windows Command Prompt" (the bottom option). Other than that, the defaults are fine. Checkout Windows-style, commit Unix-style line endings is good.

Do not forget to restart the command prompt or powershell after the installation finished successfully.

Installing Git: Debian or Ubuntu

command-line

```
$ sudo apt-get install git
```

command-line

```
$ sudo dnf install git
```

command-line

```
$ sudo zypper install git
```

Go to GitHub.com and sign up for a new, free user account.

Next it's time to sign up for a free "Beginner" account on PythonAnywhere.

- www.pythonanywhere.com

> **Note** When choosing your username here, bear in mind that your blog's URL will take the form `yourusername.pythonanywhere.com`, so choose either your own nickname, or a name for what your blog is all about.

Congratulations, you are all set up and ready to go! If you still have some time before the workshop, it would be useful to start reading a few of the beginning chapters:

- How the internet works
- Introduction to the command line

- Introduction to Python
- What is Django?

  When you begin the workshop, you'll be able to go straight to Your first Django project! because you already covered the material in the earlier chapters.

tutorial.djangogirls.org

Introduction · Django Girls Tutorial

DjangoGirls

4-5 minuti

chat on gitter

Welcome

Welcome to the Django Girls Tutorial! We are happy to see you here :) In this tutorial, we will take you on a journey under the hood of web technologies, offering you a glimpse of all the bits and pieces that need to come together to make the web work as we know it.

As with all unknown things, this is going to be an adventure - but no worries, since you already worked up the courage to be here, you'll be just fine :)

Introduction

Have you ever felt that the world is more and more about technology to which you cannot (yet) relate? Have you ever wondered how to create a website but have never had enough motivation to start? Have you ever thought that the software world is too complicated for you to even try doing something on your own?

Well, we have good news for you! Programming is not as hard as it seems and we want to show you how fun it can be.

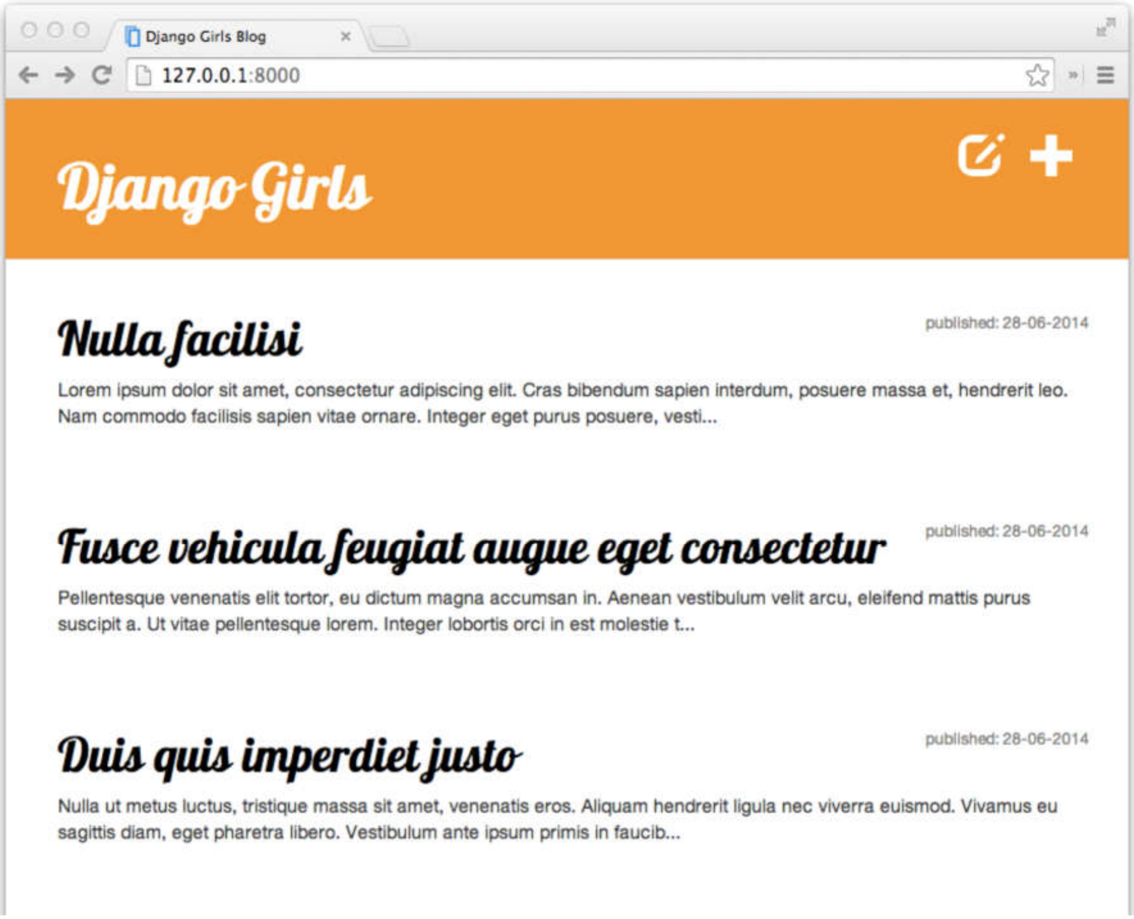
This tutorial will not magically turn you into a programmer. If you want to be good at it, you need months or even years of learning and practice. But we want to show you that programming or creating websites is not as complicated as it seems. We will try to explain different bits and pieces as well as we can, so you will not feel intimidated by technology.

We hope that we'll be able to make you love technology as much as we do!

What will you learn during the tutorial?

Once you've finished the tutorial, you will have a simple, working web application: your own blog. We will show you how to put it online, so others will see your work!

It will (more or less) look like this:



Nulla facilisi

published: 28-06-2014

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras bibendum sapien interdum, posuere massa et, hendrerit leo. Nam commodo facilisis sapien vitae ornare. Integer eget purus posuere, vesti...

Fusce vehicula feugiat augue eget consectetur

published: 28-06-2014

Pellentesque venenatis elit tortor, eu dictum magna accumsan in. Aenean vestibulum velit arcu, eleifend mattis purus suscipit a. Ut vitae pellentesque lorem. Integer lobortis orci in est molestie t...

Duis quis imperdiet justo

published: 28-06-2014

Nulla ut metus luctus, tristique massa sit amet, venenatis eros. Aliquam hendrerit ligula nec viverra euismod. Vivamus eu sagittis diam, eget pharetra libero. Vestibulum ante ipsum primis in faucib...

> If you work with the tutorial on your own and don't have a coach who will help you in case of any problem, we have a chat system for you:
> chat on gitter
> . We asked our coaches and previous attendees to be there from time to time and help others with the tutorial! Don't be afraid to ask your question there!

OK, let's start at the beginning…

Following the tutorial at home

It is amazing to take part in a Django Girls workshop, but we are aware that it is not always possible to attend one. This is why we encourage you to try following this tutorial at home. For readers at home we are currently preparing videos that will make it easier to follow the tutorial on your own. It is still a work in progress, but more and more things will be covered soon at the Coding is for girls YouTube channel.

In every chapter already covered, there is a link that points to the correct video.

About and contributing

This tutorial is maintained by DjangoGirls. If you find any mistakes or want to update the tutorial please follow the contributing guidelines.

Would you like to help us translate the tutorial to other languages?

Currently, translations are being kept on crowdin.com platform at:

https://crowdin.com/project/django-girls-tutorial

If your language is not listed on crowdin, please open a new issue informing us of the language so we can add it.

tutorial.djangogirls.org

Introduction to command line · Django Girls Tutorial

DjangoGirls

7-9 minuti

> For readers at home: this chapter is covered in the Your new friend: Command Line video.

It's exciting, right?! You'll write your first line of code in just a few minutes! :)

Let us introduce you to your first new friend: the command line!

The following steps will show you how to use the black window all hackers use. It might look a bit scary at first but really it's just a prompt waiting for commands from you.

> **Note** Please note that throughout this book we use the terms 'directory' and 'folder' interchangeably but they are one and the same thing.

What is the command line?

The window, which is usually called the **command line** or **command-line interface**, is a text-based application for viewing, handling, and manipulating files on your computer. It's much like Windows Explorer or Finder on the Mac, but without the graphical interface. Other names for the command line are: *cmd*, *CLI*, *prompt*, *console* or *terminal*.

Open the command-line interface

To start some experiments we need to open our command-line interface first.

Go to Start menu → Windows System → Command Prompt.

> On older versions of Windows, look in Start menu → All Programs → Accessories → Command Prompt.

Go to Applications → Utilities → Terminal.

It's probably under Applications → Accessories → Terminal, but that may depend on your system. If it's not there, just Google it. :)

Prompt

You now should see a white or black window that is waiting for your commands.

If you're on Mac or Linux, you probably see `$`, just like this:

command-line

```
$
```

On Windows, it's a `>` sign, like this:

command-line

```
>
```

Each command will be prepended by this sign and one space, but you don't have to type it. Your computer will do it for you. :)

> Just a small note: in your case there may be something like `C:\Users\ola>` or `Olas-MacBook-Air:~ ola$` before the prompt sign, and this is 100% OK.

The part up to and including the `$` or the `>` is called the *command line prompt*, or *prompt* for short. It prompts you to input something there.

In the tutorial, when we want you to type in a command, we will include the `$` or `>`, and occasionally more to the left. You can ignore the left part and just type in the command which starts after the prompt.

Your first command (YAY!)

Let's start with something simple. Type this command:

Your first command: OS X and Linux

Your first command: Windows

And then hit `enter`. This is our result:

command-line

```
$ whoami
olasitarska
```

As you can see, the computer has just printed your username. Neat, huh? :)

> Try to type each command; do not copy-paste. You'll remember more this way!

Basics

Each operating system has a slightly different set of commands for the command line, so make sure to follow instructions for your operating system. Let's try this, shall we?

Current directory

It'd be nice to know where are we now, right? Let's see. Type this command and hit `enter`:

Current directory: OS X and Linux

command-line

```
$ pwd
/Users/olasitarska
```

> Note: 'pwd' stands for 'print working directory'.

Current directory: Windows

command-line

```
> cd
C:\Users\olasitarska
```

> Note: 'cd' stands for 'change directory'. With powershell you can use pwd just like on Linux or Mac OS X.

You'll probably see something similar on your machine. Once you open the command line you usually start at your user's home directory.

List files and directories

So what's in it? It'd be cool to find out. Let's see:

List files and directories: OS X and Linux

command-line

```
$ ls
Applications
Desktop
Downloads
Music
...
```

List files and directories: Windows

command-line

```
> dir
 Directory of C:\Users\olasitarska
```

```
05/08/2014 07:28 PM <DIR>      Applications
05/08/2014 07:28 PM <DIR>      Desktop
05/08/2014 07:28 PM <DIR>      Downloads
05/08/2014 07:28 PM <DIR>      Music
...
```

> Note: In powershell you can also use 'ls' like on Linux and Mac OS X.

Change current directory

Now, let's go to our Desktop directory:

Change current directory: OS X and Linux
command-line

```
$ cd Desktop
```

Change current directory: Windows
command-line

```
> cd Desktop
```

Check if it's really changed:

Check if changed: OS X and Linux
command-line

```
$ pwd
/Users/olasitarska/Desktop
```

Check if changed: Windows
command-line

```
> cd
C:\Users\olasitarska\Desktop
```

Here it is!

> PRO tip: if you type `cd D` and then hit `tab` on your keyboard, the

> command line will automatically fill in the rest of the name so you can navigate faster. If there is more than one folder starting with "D", hit the `tab` key twice to get a list of options.

Create directory

How about creating a practice directory on your desktop? You can do it this way:

Create directory: OS X and Linux
command-line

```
$ mkdir practice
```

Create directory: Windows
command-line

```
> mkdir practice
```

This little command will create a folder with the name `practice` on your desktop. You can check if it's there just by looking on your Desktop or by running a `ls` or `dir` command! Try it. :)

> PRO tip: If you don't want to type the same commands over and over, try pressing the `up arrow` and `down arrow` on your keyboard to cycle through recently used commands.

Exercise!

A small challenge for you: in your newly created `practice` directory, create a directory called `test`. (Use the `cd` and `mkdir` commands.)

Solution:

Exercise solution: OS X and Linux

command-line

```
$ cd practice
$ mkdir test
$ ls
test
```

Exercise solution: Windows

command-line

```
> cd practice
> mkdir test
> dir
05/08/2014 07:28 PM <DIR>      test
```

Congrats! :)

Clean up

We don't want to leave a mess, so let's remove everything we did until that point.

First, we need to get back to Desktop:

Using `..` with the `cd` command will change your current directory to the parent directory (that is, the directory that contains your current directory).

Check where you are:

Check location: OS X and Linux

command-line

```
$ pwd
/Users/olasitarska/Desktop
```

command-line

```
> cd
```

```
C:\Users\olasitarska\Desktop
```

Now time to delete the `practice` directory:

> **Attention**: Deleting files using `del`, `rmdir` or `rm` is irrecoverable, meaning *the deleted files will be gone forever*! So be very careful with this command.

Delete directory: Windows Powershell, OS X and Linux

command-line

```
$ rm -r practice
```

Delete directory: Windows Command Prompt

command-line

```
> rmdir /S practice
practice, Are you sure <Y/N>? Y
```

Done! To be sure it's actually deleted, let's check it:

Check deletion: OS X and Linux

Exit

That's it for now! You can safely close the command line now. Let's do it the hacker way, alright? :)

Cool, huh? :)

Summary

Here is a summary of some useful commands:

Command (Windows)	Command (Mac OS / Linux)	Description	Example
exit	exit	close the window	**exit**

Command (Windows)	Command (Mac OS / Linux)	Description	Example
cd	cd	change directory	**cd test**
cd	pwd	show the current directory	**cd** (Windows) or **pwd** (Mac OS / Linux)
dir	ls	list directories/files	**dir**
copy	cp	copy file	**copy c:\test\test.txt c:\windows\test.txt**
move	mv	move file	**move c:\test\test.txt c:\windows\test.txt**
mkdir	mkdir	create a new directory	**mkdir testdirectory**
rmdir (or del)	rm	delete a file	**del c:\test\test.txt**
rmdir /S	rm -r	delete a directory	**rm -r testdirectory**

These are just a very few of the commands you can run in your command line, but you're not going to use anything more than that today.

If you're curious, ss64.com contains a complete reference of commands for all operating systems.

Ready?

Let's dive into Python!