What is Python best used for?

**Python** is a general purpose programming language created in the late 1980s, and named after Monty **Python**, that's **used** by thousands of people to do things from testing microchips at Intel, to powering Instagram, to building video games with the PyGame library.

Which companies are using python?

**Some companies I know that use python are:**

* Google (Youtube)
* Facebook (Tornado)
* Dropbox.
* Yahoo.
* NASA.
* IBM.
* Mozilla.
* Quora :D
* Instagram (django I think)
* Reddit

What is the Python programming language?

An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly braces or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as **C**++ or **Java**.

What is Python used for at Google?

At **Google**, **python** is one of the 3 "official languages" alongside with C++ and Java. Official here means that Googlers are allowed to deploy these languages to production services. (Internally **Google** people use many technologies including PHP, C#, Ruby and Perl).

Web Development using Python & Django

## LET'S GET STARTED

Welcome to learning Web development using Python & Django. Our goal is to help you understand the fundamentals well enough that you are ready to build any app you want using resources on the web. We will begin with the very basics, assuming no programming or web background. Feel free to skip sections you are already familiar with.

#### 1.1 HOW THE WEB WORKS

Before you can start building websites and web-applications, you need to understand how the web works. Along with other resources, we have hand-picked the best portions of the Udacity Web development course by the awesome Steve Huffman (founder of Reddit and Hipmunk), starting with the first lecture below. You will need to register for a free Udacity account to access this.

#### 1.2 OPTIONAL: BUILD YOUR FIRST WEBPAGE

## 2. FRONT-END

A typical web-application has two main components - the front-end and the back-end. The front-end is what the user sees - the pages, the styling and the interactive features. These are typically coded using languages like HTML and CSS. The back-end contains the logic to retrieve and process the information for the front end to display. Back-end development typically requires writing server programs in a language like PHP, Perl, Ruby or Python. Each of these is can be vast and specialized in its own right, and developers often specialize in one or the other.  
  
We cover the basics of both so that a) you get a complete understanding of the architecture of a web application b) you develop a common language that will help you work with counterparts on a real team. After completing this workshop, you could choose to go deeper in either (or both!) depending on your interests.

#### 2.1 HTML & CSS

HyperText Markup Language (HTML) is the language of the web - it allows a programmer to define the structure of a web page, interlink multiple pages and allows browsers to understand and display these pages in a consistent fashion. Cascaded Style Sheets (CSS) help you add layout and styles to your pages and make code and styles consistent and reusable.

#### 2.2 OPTIONAL: JAVASCRIPT

Javascript is what makes modern websites rich and interactive. Though knowledge of Javascript is not strictly required in an introductory web development course, not knowing it would be a severe limitation. Let’s take a look at the basics of Javascript and jQuery -- a very popular library.

## 3. BACK-END

To refresh your memory, back end is the logic that runs on servers to retrieve and process data for the front end to display. This data could be stored in and retrieved from databases, or could be fetched through 3rd party APIs. The communication is through a server-side language like Python, Ruby or PHP. This section will touch upon the basics of each of these components.

#### 3.1 LANGUAGE - PYTHON

There are many server-side languages to choose from. I’ve chosen Python because it’s great for beginners (you actually feel like [this](http://imgs.xkcd.com/comics/python.png) on day 1!). Also, the elegance of Python makes the code immensely readable - coding in Python might be as close as one can get to conversing with the computer in English. It is also designed for scalability and is popular in large data-crunching applications. Dropbox, Quora, Reddit and YouTube are all built in Python.

#### 3.2 DATABASES

Databases store all the information on your website. For simpler websites, it is possible to have a bunch of static HTML, CSS, and media files that do not interact with a database to fetch its content. But most modern web applications contain large scale, dynamic and often user-generated content which can not be maintained without a database. E.g. if you’re creating a social network like Facebook, you will need a database to store all the user profile & authentication information, their friends, messages, likes, posts etc.

#### 3.3 USER ACCOUNTS AND SECURITY

#### 3.4 APIS

Making a rich web application today is much easier than it would be if you had to build all the pieces yourself, thanks to Application Programming Interfaces or APIs. There are thousands of [awesome APIs](http://www.programmableweb.com/) out there which will allow you to add on top of data and functionality other developers have built.

#### 3.5 THE DJANGO FRAMEWORK

You’ve learned an awful lot in the past few sections. Get ready to put it all together and build your first awesome website! Enter Web application frameworks.

Web application frameworks support and speed up web development by providing libraries for many common tasks associated with web apps. Django was the natural choice given the power of Python. Amongst other alternatives, the Rails framework is popular as it has a very mature community and has a lot of gems (pluggable modules). But the Django community is catching up and finally the choice boils down to what you are more comfortable with. For more of the holy Django vs. Rails debate, read these threads on [Stack Overflow](http://stackoverflow.com/questions/846231/django-or-ruby-on-rails/846266#846266) and [Quora](http://www.quora.com/Ruby-vs-Python/Which-should-I-learn-Django-or-Rails" \t "_blank).

## 4. DEVELOPER TOOLS

#### 4. 1 DEBUGGING & OPTIMIZATION

Armed with an arsenal of knowledge, you are raring to build your dream app. But code rarely works in the first try. How do you troubleshoot when something breaks? How do you measure the performance of your website and make it faster? Most browsers have built-in capabilities or extensions called developer tools.

#### 4.2 GIT - VERSION CONTROL

Rome wasn’t built in a day, or by one man. The same holds true for any software of any reasonable size or significance. Version control systems help teams work together on complex software without mucking up another person’s code or losing history.

Git is the de-facto standard for version control today. It has a steep learning curve and can be intimidating in the beginning, especially to the users who are used to the simplicity of older version control systems like cvs and svn. But hang in there! Once you overcome that barrier, you will wonder how anyone ever worked without it. It is especially useful for working on multiple features in parallel and allows switching between multiple work-in-progress versions seamlessly.