**CHAPTER 1**

**INTRODUCTION**

* 1. **PYTHON:**

****

Fig.1.1 Python Logo

### Python Language Introduction:

[Python](https://www.geeksforgeeks.org/python-programming-language/) is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

**Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

### 1.1.2 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.



Fig.1.2 Guido van Rossum

### 1.1.3 Python Features

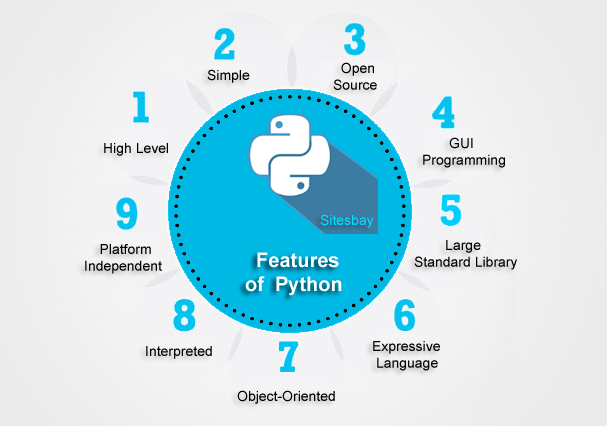


Fig 1.3 Features

Python’s Features includes-

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

**1.1.4 Future Technologies Counting On Python**

Generally, we have seen that python programming language is extensively used for web development, application development, system administration, developing games etc.

But there are some future technologies that are relying on python. As a matter of fact, Python has become the core language as far as the success of these technologies is concerned. Let’s dive into the technologies which use python as a core element for research, production and further developments.

1. **Artificial Intelligence (AI)**

Python programming language is undoubtedly dominating the other languages when future technologies like Artificial Intelligence(AI) comes into the play.

There are plenty of python frameworks, libraries, and tools that are specifically developed to direct Artificial Intelligence to reduce human efforts with increased accuracy and efficiency for various development purposes.

It is only the Artificial Intelligence that has made it possible to develop speech recognition system, autonomous cars, interpreting data like images, videos etc.

We have shown below some of the python libraries and tools used in various Artificial Intelligence branches.

Machine Learning- PyML, PyBrain, scikit-learn, MDP Toolkit, GraphLab Create, MIPy etc.

General AI- pyDatalog, AIMA, EasyAI, SimpleAI etc.

Neural Networks- PyAnn, pyrenn, ffnet, neurolab etc.

Natural Language & Text Processing- Quepy, NLTK, gensim

**(2) Big Data**

The future scope of python programming language can also be predicted by the way it has helped big data technology to grow. Python has been successfully contributing in analyzing a large number of data sets across computer clusters through its high-performance toolkits and libraries.

Let’s have a look at the python libraries and toolkits used for Data analysis and handling other big data issues.

Pandas

Scikit-Learn

NumPy

SciPy

GraphLab Create

IPython

Bokeh

Agate

PySpark

Dask

**(3) Networking**

Networking is another field in which python has a brighter scope in the future. Python programming language is used to read, write and configure routers and switches and perform other networking automation tasks in a cost-effective and secure manner.

For these purposes, there are many libraries and tools that are built on the top of the python language. Here we have listed some of these python libraries and tools especially used by network engineers for network automation.

Ansible

Netmiko

NAPALM(Network Automation and Programmability Abstraction Layer with Multivendor Support)

Pyeapi

Junos PyEZ

PySNMP

Paramiko SSH

Websites Developed Using Python

Python programming language is used for web development, so here are some of the world’s most popular websites that are created using python.

Youtube

Quora

Instagram

Pinterest

Spotify

Flipkart

Uber

Google

Facebook

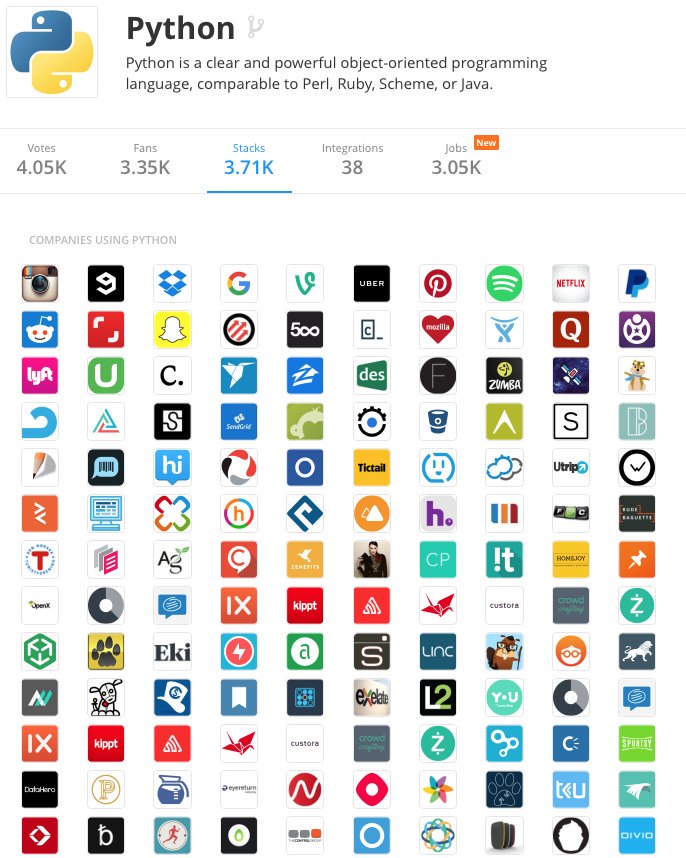


Fig 1.4 Companies Using Python

**Why Python Programming Language Has Bright Future?**

Python has been voted as most favourite programming language beating C, C++ and java programming. Python programming is open source programming language and used to develop almost every kind of application.

Python is being used worldwide as a wide range of application development and system development programming language. Big brands and search engine giants are using python programming to make their task easier. Google, Yahoo, Quora, Facebook are using python programming to solve their complex programming problems.

Python programming is versatile, robust and comprehensive. Python is high-level programming language and easy to learn as well as it reduces the coding effort compare to other programming languages.

Python programming is used to write test scripts and tests mobile devices performance. It is one of the most versatile languages these days. Python programmers are most demandable in the IT industry these days and get paid more compared to another language programmer.

# 1.2 Python - Modules

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

## Example

The Python code for a module named *a name* normally resides in a file named *aname.py*. Here's an example of a simple module, support.py

def print\_func( par ):

print"Hello : ", par

return

## The *import* Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax −

import module1[, module2[,... module N]

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. For example, to import the module support.py, you need to put the following command at the top of the script −

#!/usr/bin/python

# Import module support

import support

# Now you can call defined function that module as follows

support.print\_func("Jerry")

When the above code is executed, it produces the following result −

Hello : Jerry

A module is loaded only once, regardless of the number of times it is imported. This prevents the module execution from happening over and over again if multiple imports occur.

## The *from...import* Statement

Python's *from* statement lets you import specific attributes from a module into the current namespace. The *from...import* has the following syntax −

from modname import name1[, name2[, ... nameN]]

For example, to import the function fibonacci from the module fib, use the following statement −

from fib import fibonacci

This statement does not import the entire module fib into the current namespace; it just introduces the item fibonacci from the module fib into the global symbol table of the importing module.

## The *from...import \** Statement

It is also possible to import all names from a module into the current namespace by using the following import statement

**1.3 PYTHON-FLASK**

Flask is a popular Python web framework, meaning it is a third-party Python library used for developing web applications.

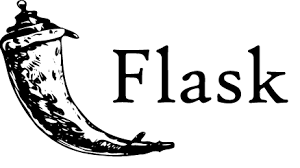
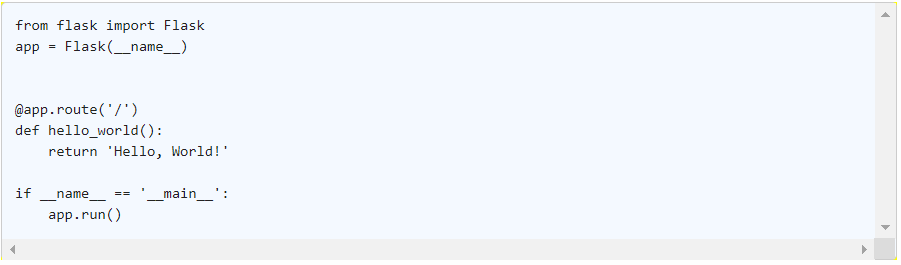


Fig 1.4 Python Flask

Flask is considered more [Pythonic](http://blog.startifact.com/posts/older/what-is-pythonic.html) than the [Django](https://www.fullstackpython.com/django.html) web framework because in common situations the equivalent Flask web application is more explicit. Flask is also easy to get started with as a beginner because there is little boilerplate code for getting a simple app up and running.

For example, here is a valid "Hello, world!" web application with Flask:



The above code shows "Hello, World!" on localhost port 5000 in a web browser when run with the python app.py command and the Flask library installed.

The equivalent "Hello, World!" web application using the [Django](https://www.fullstackpython.com/django.html) [web framework](https://www.fullstackpython.com/web-frameworks.html) would involve significantly more boilerplate code.

Flask was also written several years after Django and therefore learned from the Python community's reactions as the framework evolved. Jökull Sólberg wrote a great piece articulating to this effect in his [experience switching between Flask and Django](http://web.archive.org/web/20160305145017/http://jokull.calepin.co/my-flask-to-django-experience.html)

**1.3.1 FLASK INSTALLATION**

We recommend using the latest version of Python 3. Flask supports Python 3.3 and newer, Python 2.6 and newer, and PyPy.

These distributions will be installed automatically when installing Flask.

[Werkzeug](http://werkzeug.pocoo.org/) implementsWSGI, the standard Python interface between applica-tions and servers.

[Jinja](http://jinja.pocoo.org/) isa template language that renders the pages your application serves.

[MarkupSafe](https://pypi.python.org/pypi/MarkupSafe) comeswith Jinja. It escapes untrusted input when rendering tem-plates to avoid injection attacks.

[ItsDangerous](https://pythonhosted.org/itsdangerous/) securelysigns data to ensure its integrity. This is used to protectFlask’s session cookie.

[Click](http://click.pocoo.org/) isa framework for writing command line applications. It provides theflask command and allows adding custom management commands.

[Blinker](https://pythonhosted.org/blinker/) providessupport forSignals.

[SimpleJSON](https://simplejson.readthedocs.io/) isa fast JSON implementation that is compatible with Python’sjsonmodule. It is preferred for JSON operations if it is installed.

[python-dotenv](https://github.com/theskumar/python-dotenv#readme) enablessupport forLoading Environment Variables From .env Fileswhen running flask commands.

**Virtual environments**

Use a virtual environment to manage the dependencies for your project, both in de-velopment and in production.

What problem does a virtual environment solve? The more Python projects you have, the more likely it is that you need to work with different versions of Python libraries, or even Python itself. Newer versions of libraries for one project can break compatibility in another project.

Virtual environments are independent groups of Python libraries, one for each project. Packages installed for one project will not affect other projects or the operating sys-tem’s packages.

Python 3 comes bundled with the [venv](https://docs.python.org/3/library/venv.html#module-venv)module to create virtual environments. If you’re using a modern version of Python, you can continue on to the next section.

If you’re using Python 2, see Install virtualenv first.

**1.4 TEMPLATES**

Flask leverages Jinja2 as template engine. You are obviously free to use a different tem-plate engine, but you still have to install Jinja2 to run Flask itself. This requirement is necessary to enable rich extensions. An extension can depend on Jinja2 being present.

This section only gives a very quick introduction into how Jinja2 is integrated into Flask. If you want information on the template engine’s syntax itself, head over to the official [Jinja2 Template Documentation](http://jinja.pocoo.org/docs/templates) for more information.

**Jinja Setup**

Unless customized, Jinja2 is configured by Flask as follows:

1. autoescaping is enabled for all templates ending in .html, .htm, .xml as well as

.xhtml when using render\_template().

1. autoescaping is enabled for all strings when using render\_template\_string().
2. a template has the ability to opt in/out autoescaping with the {% autoescape %} tag.
3. Flask inserts a couple of global functions and helpers into the Jinja2 context, ad-ditionally to the values that are present by default.

**Standard Context**

The following global variables are available within Jinja2 templates by default:

**config**

The current configuration object (flask.config)

New in version 0.6.

Changed in version 0.10: This is now always available, even in imported tem-plates.

**request**

The current request object (flask.request). This variable is unavailable if the template was rendered without an active request context.

**session**

The current session object (flask.session). This variable is unavailable if the template was rendered without an active request context.

**g**

The request-bound object for global variables (flask.g). This variable is unavail-able if the template was rendered without an active request context.

**url\_for**()

The flask.url\_for() function.

**get\_flashed\_messages**()

The flask.get\_flashed\_messages() function.

**1.5 DATABASES**

A database stores application data in an organized way. The application then issues queries to retrieve specific portions as they are needed. The most commonly used databases for web applications are those based on the relational model, also called SQL databases in reference to the Structured Query Language they use. But in recent years document-oriented and key-value databases, informally known together as NoSQL databases, have become popular alternatives.

**1.5.1 SQL Databases**

Relational databases store data in tables, which model the different entities in the ap‐ plication’s domain. For example, a database for an order management application will likely have customers, products, and orders tables. A table has a fixed number of columns and a variable number of rows. The columns define the data attributes of the entity represented by the table. For example, a customers table will have columns such as name, address, phone, and so on. Each row in a table defines an actual data element that consists of values for all the columns. Tables have a special column called the primary key, which holds a unique identifier for each row stored in the table. Tables can also have columns called foreign keys, which reference the primary key of another row from the same or another table. These links between rows are called relationships and are the foundation of the relational database model.

**1.5.2 Python Database Frameworks**

Python has packages for most database engines, both open source and commercial. Flask puts no restrictions on what database packages can be used, so you can work with MySQL, Postgres, SQLite, Redis, MongoDB, or CouchDB if any of these is your favorite. As if those weren’t enough choices, there are also a number of database abstraction layer packages such as SQLAlchemy or MongoEngine that allow you to work at a higher level with regular Python objects instead of database entities such as tables, documents, or query languages. There are a number of factors to evaluate when choosing a database framework: Ease of use When comparing straight database engines versus database abstraction layers, the second group clearly wins. Abstraction layers, also called object-relational mappers (ORMs) or object-document mappers (ODMs), provide transparent conversion of high-level object-oriented operations into low-level database instructions.

**1.6**

Flask was originally designed and developed by Armin Ronacher as an [April Fool's Day joke in 2010](http://lucumr.pocoo.org/2010/4/3/april-1st-post-mortem/). Despite the origin as a joke, the Flask framework became wildly popular as an alternative to Django projects with their monolithic structure and dependencies.

Flask's success created a lot of additional work in issue tickets and pull requests. Armin eventually created [The Pallets Projects](https://www.palletsprojects.com/) collection of open source code libraries after he had been managing Flask under his own GitHub account for several years. The Pallets Project now serves as the community-driven organization that handles Flask and other related Python libraries such as [Lektor](https://www.fullstackpython.com/lektor.html), [Jinja](https://www.fullstackpython.com/jinja2.html) and several others.

**1.7 Flask vs Django**

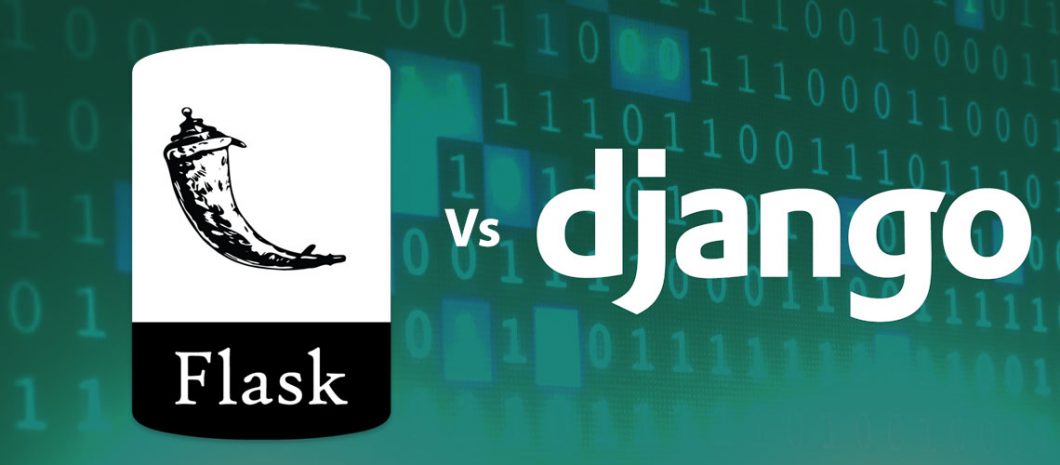


Fig.1.5 Comparision

A web developer has option to choose from a wide range of web frameworks while using Python as server-side programming languages. He can take advantage of the full-stack Python web frameworks to accelerate development of large and complex web applications by availing a number of robust features and tools. Likewise, he can also opt for micro and lightweight Python web frameworks to build simple web applications without putting extra time and effort.

Both Django and Flask are hugely popular among Python programmers. Django is a full-stack web framework for Python, whereas Flask is a lightweight and extensible Python web framework. Django is developed based on batteries-included approach. It enables programmers to accomplish common web development tasks without using third-party tools and libraries. Django lacks some of the robust features provided by Python.

But it is developed based on two robust POCO projects – Web Server Gateway Interface (WSGI) toolkit and Jinja2 template engine. The developers can take advantage of WSGI to communicate web application and web server through a universal interface. Likewise, Jinja2 template engine makes it easier for programmers to build dynamic web applications. Hence, it becomes essential for web developers to understand the major differences between Flask vs Django.

Flask is more Pythonic than Django because the code of flask Web Application in most cases is more explicit than the Django Web Application code. So it is easy for Python coders to pick up.

Beginners can learn a lot from the well documented Source Code.

Flask does include everything you need to start building something more complex than the above Hello World app. It integrates a templating engine (Jinja2) for you so you don't have to decide whether you would be better off using Genshi, Cheetah or Mako (though you could use).

The extra work at start to choose components for Flask apps yields more flexibility for developers whose use case doesn't fit a standard ORM, or who need to interoperate with different workflows or templating systems.

Django is a large framework which includes everything, whether you require it or not. So for smaller application Flask can give more performance. [httpbin](https://httpbin.org/) is a smaller application built with Flask.

Flask also scales well. Pinterest uses Flask for their API. The API Handles over 12 billion requests per day, all in Flask.

**CHAPTER 2**

**TOOLS AND TECHNIQUES**

**2.1 PLATFORM USED:**

This section is to present a detailed description of the Platforms used for this project. It explains the hardware and software requirements for developing the application and its interface, tested features of the program,

* what the program will do,
* the constraints under which it must operate
* and how the program react to external stimuli.

**2.1.1 Hardware Used:**

This program has been developed using 4 GB of RAM, i5 1.70 GHz 1 TB Hard disk space but the Minimum requirement of such kind of application for a machine is:

1 GHz or faster processor 1 GB of RAM (1.5 GB if running on a virtual machine)

10 GB of available hard disk space 5400 RPM hard drive

DirectX 9-capable video card running at 1024 x 768 or higher display resolution.

**2.1.2 Software Used:**

This program is developed in a platform of Microsoft windows 10 Operating System. Microsoft Windows is a series of graphical interface operating systems designed, developed, marketed, and sold by Microsoft onwards from November 20, 1985. The most recent versions of Windows are windows 10 and 10.1.The Other Supported Operating systems are:

Windows 7 SP1 (x86 and x64)

Windows 8 (x86 and x64)

Windows 10 (x64 and x86)

**There are also many ide’s (integrated development environment )available for Python. A brief list is as shown below:**

|  |  |  |  |
| --- | --- | --- | --- |
| **IDE** | **Type** | **Platform** | **Url** |
| **IDLE** – A GUI based IDE that comes pre-installed with the reference implementation | Open Source | Windows, Mac OS, Linux | https://www.python.org/download / |
| **PyCharm** – A GUI based IDE by Jetbrains supporting code-completion, code-debugging, refactoring etc. | Commercial/Free Community Edition | Windows, Mac OS, Linux | http://www.jetbrains.com/pychar m/download/ |
| **Anaconda** is a free and open source distribution of the Python and R programming languages for data science and machine learning related applications that aims to simplify package management and deployment | Free and Open Source | Windows, Mac OS, Linux | https://www.anaconda.com/download/ |

In this Project we have used PyCharm as an integrated development environment.



Fig1.6 PyCharm

**PyCharm** is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) used in [computer programming](https://en.wikipedia.org/wiki/Computer_programming), specifically for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) language. It is developed by the Czech company [JetBrains](https://en.wikipedia.org/wiki/JetBrains).PyCharm Community Edition is totally free and open-source, available under the Apache 2.0 license.

PyCharm makes it easier for programmers to write various web applications in Python supporting widely used web technologies like HTML, CSS, JavaScript, TypeScript and CoffeeScript

PyCharm is available in three editions: Professional, Community, and Educational (Edu). The Community and Edu editions are open-source projects and they are free, but they have less features. PyCharm Edu provides courses and helps you learn programming with Python.

**CHAPTER 3**

**SNAPSHOTS**

**SCREENSHOTS:**

1. **Home page.**

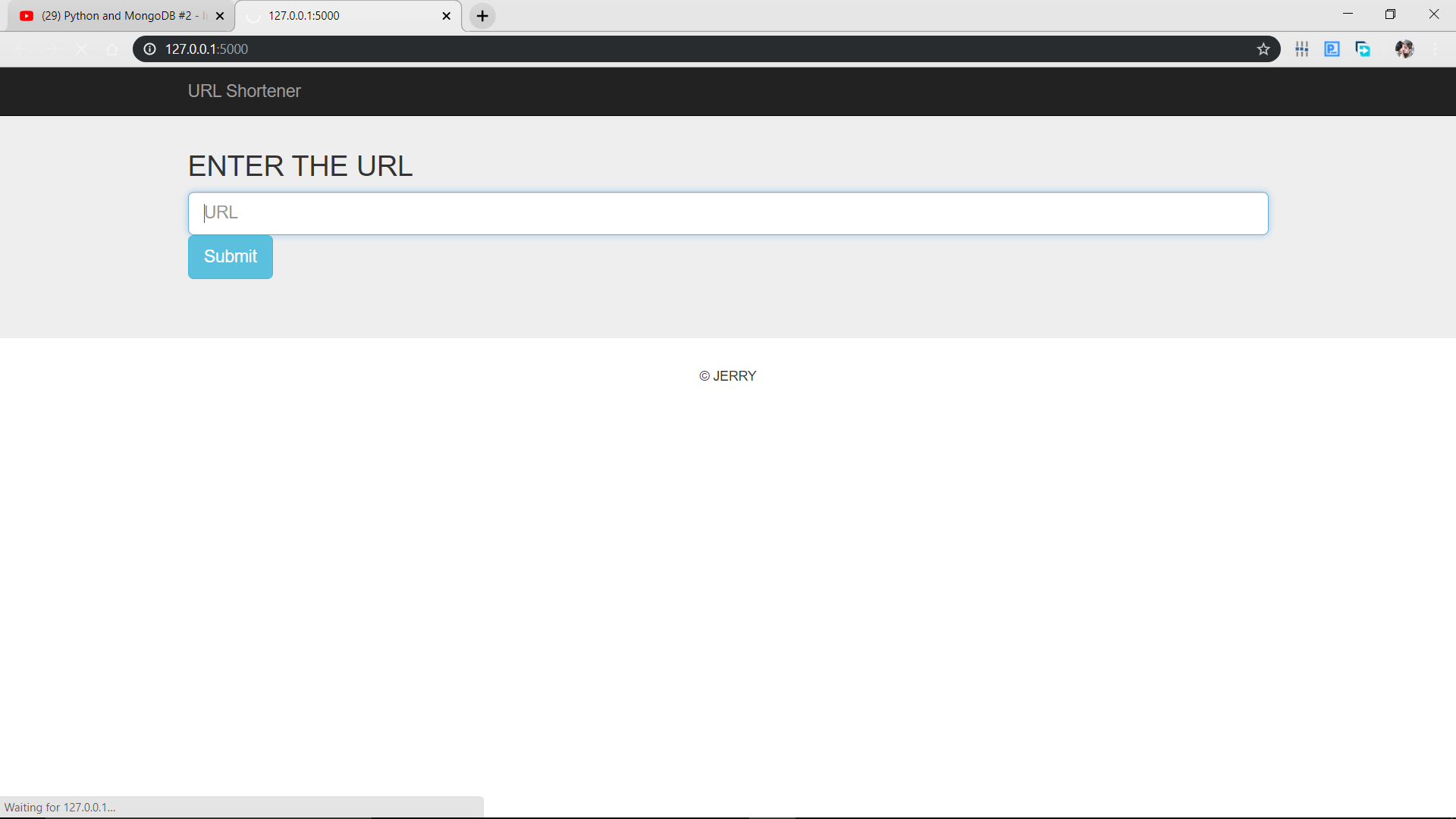
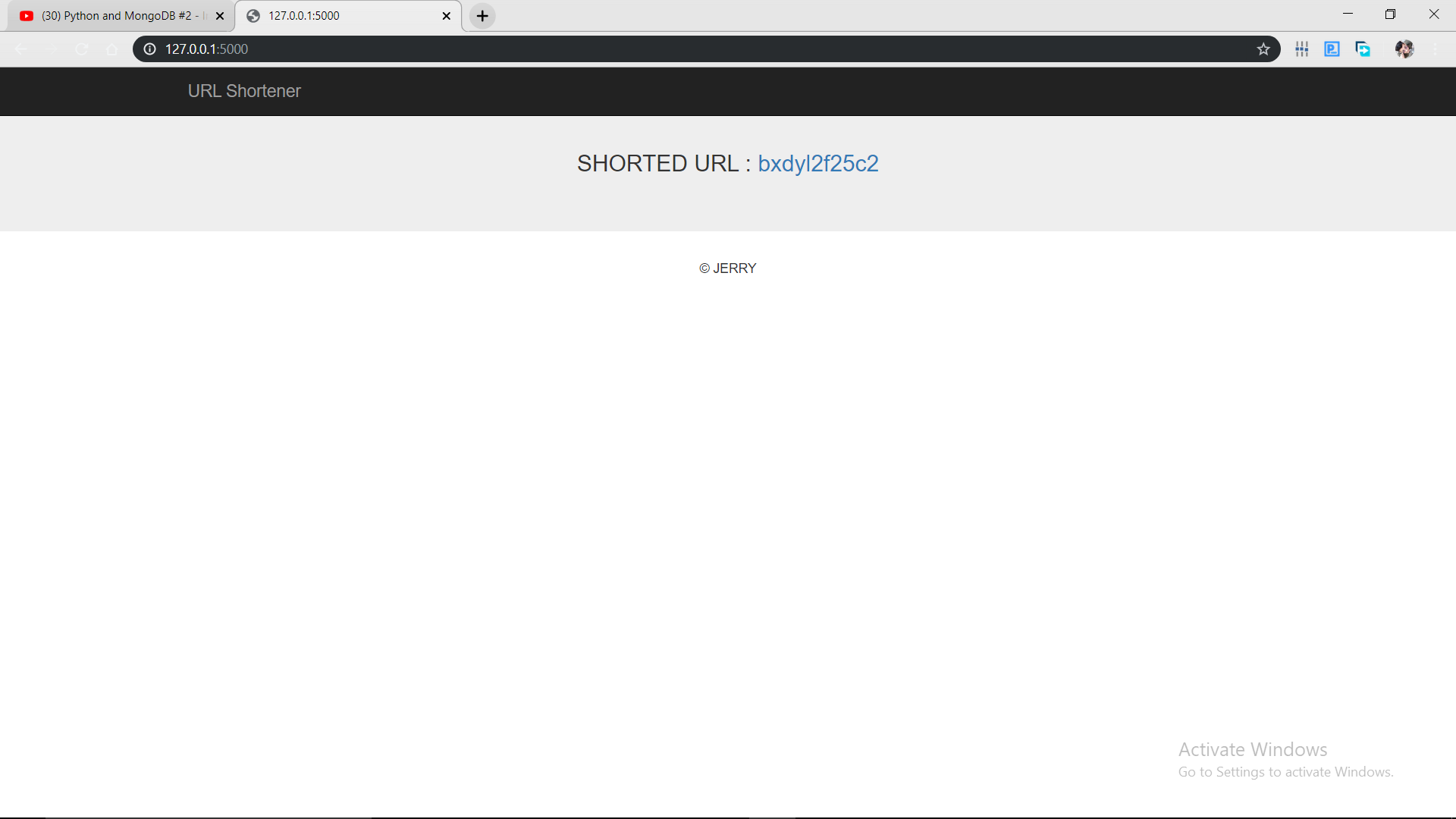


Fig 1.7 shortening Page

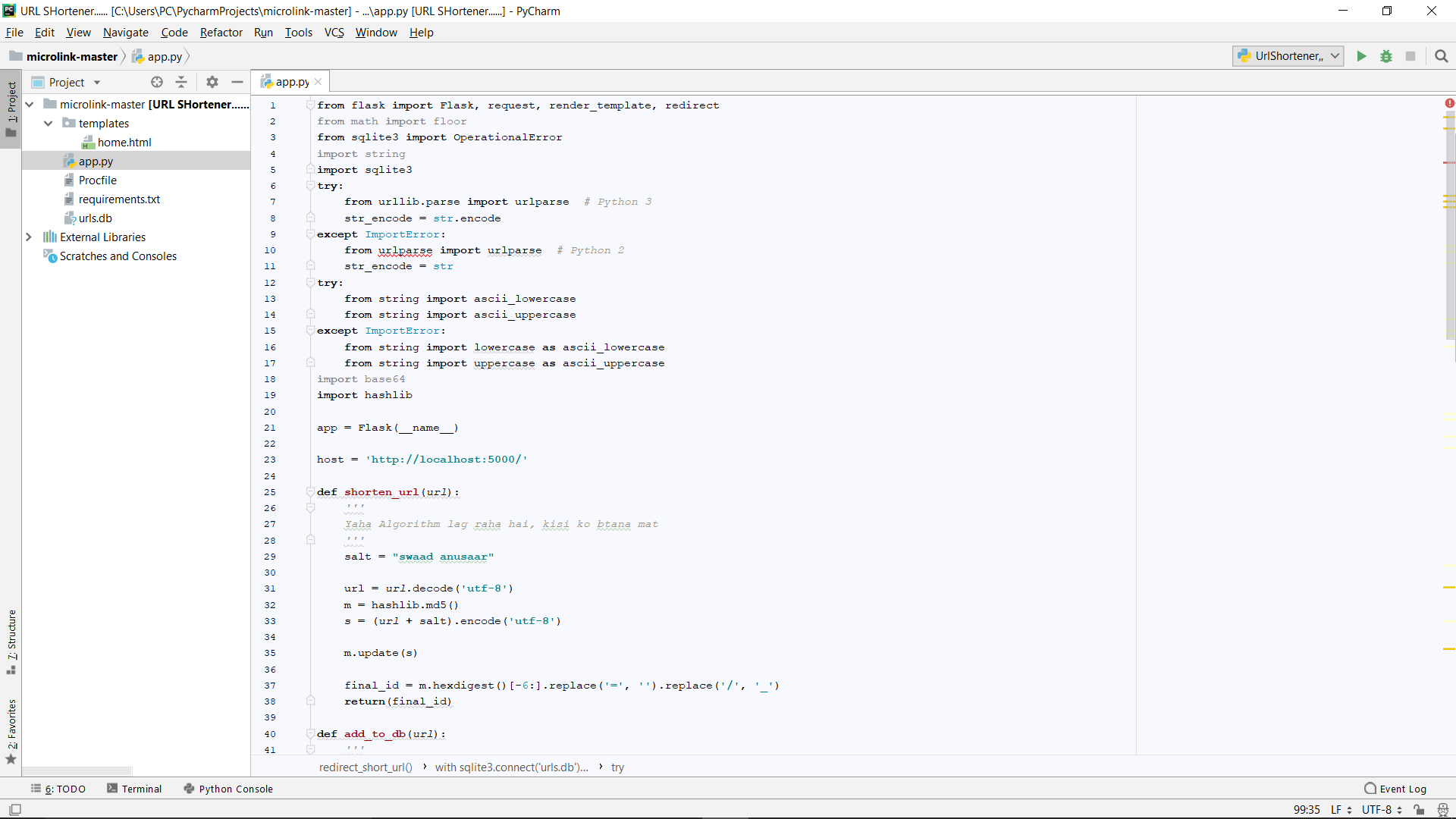
**127.0.0.1:5000** is Local Host Address used for Web Applications.

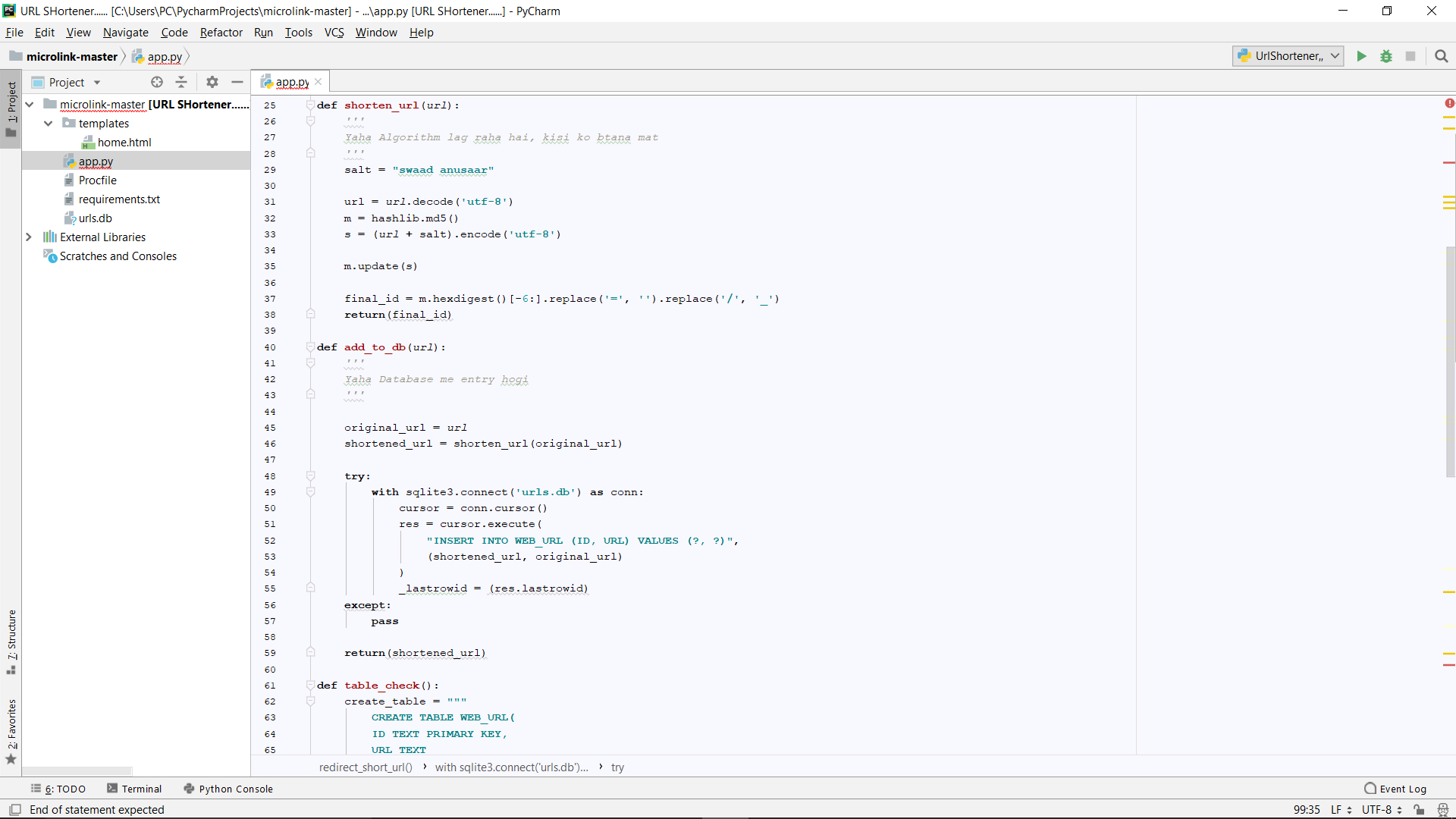
1. **Shorted URL**

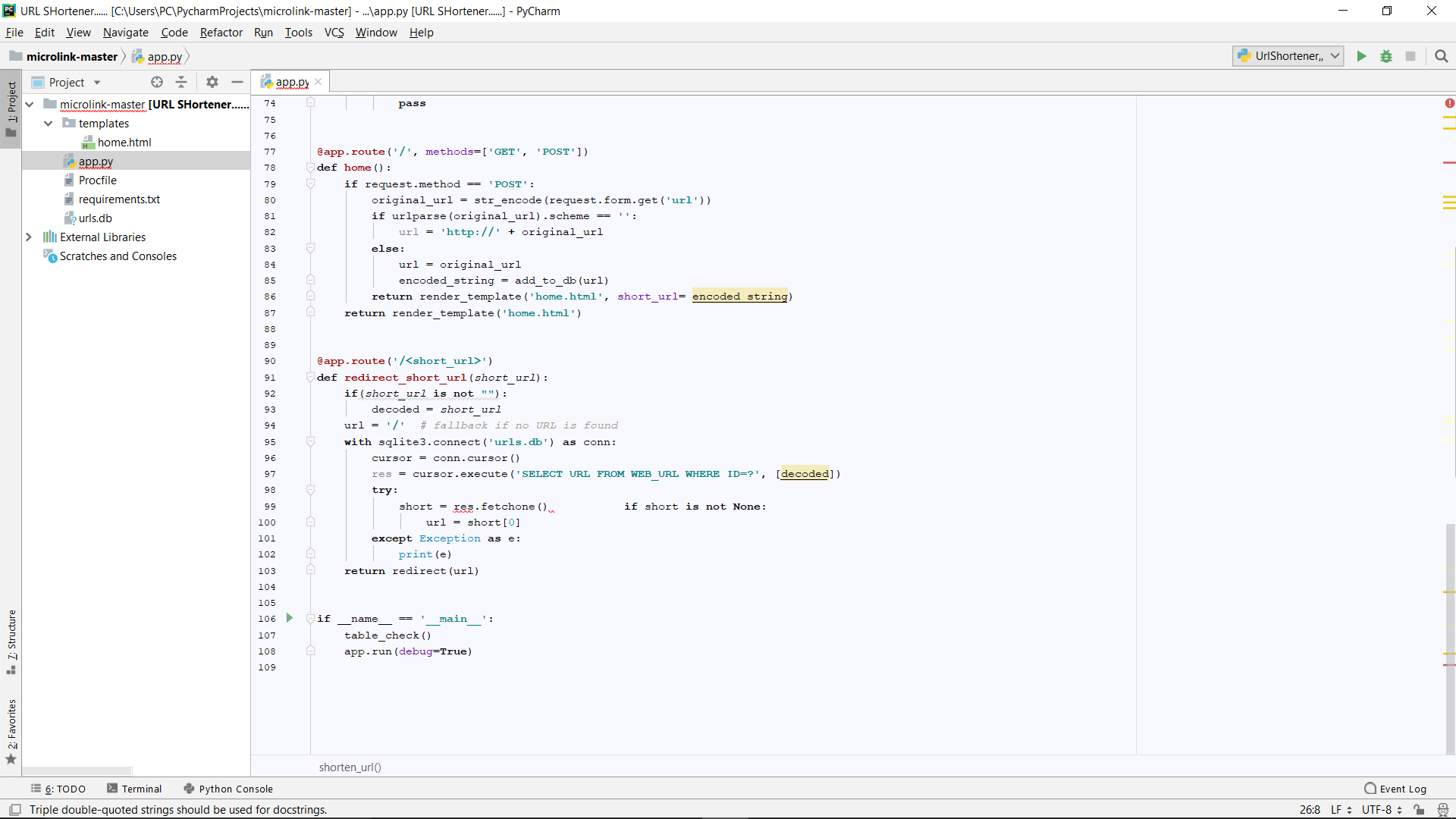
****

**Fig 1.8 Shorted URL**

1. **Code Screenshot:**

****

****

****

**CHAPTER 4**

**RESULT AND DISCUSSION**

URL Shortener is a feature that allows you to create short URLs for any page on projects hosted by the Wikimedia Foundation, in order to reuse them elsewhere, for example on social networks, on wikis, or on paper.

The feature can be accessed from Meta wiki on the special page [m:Special:URLShortener](https://meta.wikimedia.org/wiki/Special:URLShortener). On this page, you will be able to enter any web address from a service hosted by the Wikimedia Foundation, to generate a short URL, and to copy it and reuse it anywhere.

The format of the URL is w.wiki/ followed by a string of letters and numbers. You can already test this, for example:

<https://w.wiki/3> redirects to wikimedia.org

**4.1 Advantages of Url Shortener**

## 1. They make links more manageable

No, not all of the web's links are that messy. However, many links are long and wordy for SEO reasons. SEO, or search engine optimization, is about ranking higher in [Google](https://mashable.com/category/google/), and one of the factors Google and other search engines consider are keywords in the URL. This creates a conundrum for the user - the URLs help describe the content, but are lengthy and are not easy to share on emails, web pages, and especially social media services like [Facebook](https://mashable.com/category/facebook/) and Twitter.

URL shorteners help solve the problem of making links more manageable to share. And certain services, such as [➡.ws](http://tinyarro.ws/) allows users to add keywords to describe the link.

## 2. They can track and compile click data

One of the reasons that it has received so much attention lately is because of the comprehensive data Bitly provides in the form of live click data, geographic location, the webpage the link where the link was clicked, and more. This type of information is invaluable to webmasters and companies - it shows where customers are coming from, when they are coming, and what interests them.

This type of information helps companies develop better products and webmasters produce more targeted content. Detailed information makes the economy more efficient.

## 3. They can be transformed into social media services

There was a good piece on GigaOM that discussed how Bit.ly could [launch its own version of Digg](http://gigaom.com/2009/03/31/why-bitly-could-upstage-digg/). While this may or may not be Bit.ly's eventual goal (just like Digg, a URL shortener could be gamed), it's clear that the data that URL shorteners can accumulate, coupled with the rise of short URL sharing on Twitter and other websites, could amount to some innovative social media services that display popular links, rank domains, and act as a filter or aggregate of social media content.

URL shorteners, in their own way, work as aggregates of information. This can lead to some useful mashups and innovations in how people share and digest content.

## 4. They can provide users useful features

A new URL shortener, Pagetweet, caught my eye. It's a little more complicated than other URL shorteners (I don't understand why it requires a security code), but when you look at an actual page via Pagetweet, it provides a useful interface for sharing via social media, seeing the number of views, and more.

Because it's so easy for companies to enter this space, innovations are constantly being made that improve the user experience. [Digg](https://mashable.com/category/digg/) recently launched a [DiggBar and Digg URL shortener](https://mashable.com/2009/04/02/diggbar-shorten-urls-and-experience-digg-on-any-web-page/), which provides information on the number of Diggs and comments any article has received. This is only the beginning.

Users also have a choice - they can follow links that will provide features, or simply choose ones with only the necessary functionality to get them to a web page. All in all though, URL shorteners can improve the browsing experience.

## 5. They promote sharing

You can simply fit more links and content in less space with URL shorteners. A tweet can describe and then link to a webpage in under 140 characters, while a full URL might not even come with an explanation.

Even more important is the rise of mobile smartphones, texting, and mobile Internet - it's far easier to text in a short URL than a long one. As Twitter, social media, and mobile Internet become more popular, the need to make sharing web content easier will increase. Shorter URLs are becoming more and more integral to that cause.

**4.2 Disadvantages of url shortener**

**1. Short URLs can be difficult to memorize.**  
Even though they are short in length, these shortened URLs may still be a series of random characters similar to what you’ll see in a typical YouTube URL. Although some services have custom alias option, your chosen vanity URL could also be unavailable, like your favorite domain names snapped by someone else long time ago. You don’t need to memorize them, but if this short URL appears on offline channels, its ability to bridge between offline (magazine commercials, billboard ads or direct mail invitations) and your destination URL depends on how easy for your audience to remember them, or will they be put off or misguided by the sight of some random character elements.

1. **Third-party short URL service can shut down.**

If Geocities and Google Answers got shut down, so will these third-party URL shortening services, especially if they are not profitable nor getting financial support. We all hope this won’t happen but if it does, all the your links you connected with this service will not work. Not even a 301 redirection will solve the issue. So that’s a risk when you created these short URLs and placed them on offline channels for people to view and access, suddenly they became broken links. Another major impact is that your destination URL will also lose a portion of its inbound links that may or may not impact your SEO efforts.

1. **Short URL offers no clue to destination URL.**

Every single URL generated by the URL shortening services like Bit.ly, Owl.ly, TinyURL and Su.pr will show no clue about the destination URL. Wait, there’s a little clue if the alias used provides descriptive hint, though I don’t expect it to be very descriptive otherwise it might be too long that it’s no longer a short URL. With this setup people might have hesitation on whether to click this link or not.

1. **URL shortener service can go offline.**

Unlike item #2 which is a permanent shutdown of service and all links you generate are rendered useless, URL services can suffer intermittent disruption of service. Power supply issues, security intrusion or any forms of cyber attack could result in links that don’t work. If you are running an important campaign during the same period and you are using this particular service, the success of your campaign might hang in the balance. Your success could depend on how this service secured its infrastructure. If it fails, your brand likely cannot avoid negative impression.

1. **Short URLs can lead to malicious, unsafe web pages.**

Since the destination landing page URL is not visible when it is shortened, people will have no idea where they’ll land on once they click the link or type the short URL address. Even if provided with a hint, that might just be a disguise to attract audience and lead them to phishing sites or those that pose security issues such as those that auto-download malicious programs that extract information from your machine or phone.

**6. Generic short URLs mask brand identity.**

If you are a big organization, part of your branding exercise should include even the URL shortening service. Using generic short URLs don’t reveal your brand identity. Pepsi (pep.si), Virgin (virg.in), BBC (bbc.in) and The New York Times (nyti.ms) have their own custom link shortening service when sharing content on social media. Using generic short URLs isn’t necessarily bad (discounting all the disadvantages I listed here) but having your own custom URL can improve brand visibility and consistency.

**7. Short URL services likely slow speed of the web.**

By using an extra layer of URL before forwarding to a destination landing page, there exists an extra step in the journey of a visitor to a website. But even though 301 redirection method is supposed to be quick and painless, speed could be an issue when bridging two sites of different technical backgrounds. [A study in 2010](https://techcrunch.com/2010/03/17/url-shorteners-speed/) showed that all of the 14 URL shortening services tested have significantly slowed down page loading time. Seven years later, we can only hope these are already resolved. Otherwise, this page loading speed issue which is one of search engine factors, might impact search visibility of the landing page.

**CHAPTER 5**

**CONCLUSIONS AND FUTURE SCOPE**

* 1. **CONCLUSION**

This project work is an attempt to develop web application which can be used to shorten the URL’s.

The main feature of our project is that it is time saving, as there are very long URLs which are hard to remember. Also the software used in this project, python is a flexible language which can run in any type of system. It is user friendly and very easy to use.

URL shorteners offer a range of pros and cons, and it can be difficult to know when it’s best to make use of them. However, knowledge is power, and studying the positives and the negatives will offer some insight into whether they’re right for you.

URL Shortener specifically helps to mitigate the negatives of link shortening by combining good aesthetics with powerful social sharing, 301 permanent redirects, and robust analytics. Also, unlike so many URL shortening services, Pretty Links enables you to bolster your branding efforts through the retention of your custom domain.

This project has really been faithful and informative. It has made us learn and understand the many trivial concepts of Python Language.Even more important is the rise of mobile smartphones, texting, and mobile Internet - it's far easier to text in a short URL than a long one. As Twitter, social media, and mobile Internet become more popular, the need to make sharing web content easier will increase. Shorter URLs are becoming more and more integral to that cause.

Every single URL generated by the URL shortening services like Bit.ly, Owl.ly, TinyURL and Su.pr will show no clue about the destination URL. Wait, there’s a little clue if the alias used provides descriptive hint, though I don’t expect it to be very descriptive otherwise it might be too long that it’s no longer a short URL. With this setup people might have hesitation on whether to click this link or not.

One of the reasons that it has received so much attention lately is because of the comprehensive data Bitly provides in the form of live click data, geographic location, the webpage the link where the link was clicked, and more. This type of information is invaluable to webmasters and companies - it shows where customers are coming from, when they are coming, and what interests them.

Flask is considered more [Pythonic](http://blog.startifact.com/posts/older/what-is-pythonic.html) than the [Django](https://www.fullstackpython.com/django.html) web framework because in common situations the equivalent Flask web application is more explicit. Flask is also easy to get started with as a beginner because there is little boilerplate code for getting a simple app up and running.

Flask is more Pythonic than Django because the code of flask Web Application in most cases is more explicit than the Django Web Application code. So it is easy for Python coders to pick up.Beginners can learn a lot from the well documented Source Code.

Flask does include everything you need to start building something more complex than the above Hello World app. It integrates a templating engine (Jinja2) for you so you don't have to decide whether you would be better off using Genshi, Cheetah or Mako (though you could use).

The extra work at start to choose components for Flask apps yields more flexibility for developers whose use case doesn't fit a standard ORM, or who need to interoperate with different workflows or templating systems.

Django is a large framework which includes everything, whether you require it or not. So for smaller application Flask can give more performance. [httpbin](https://httpbin.org/) is a smaller application built with Flask.

* 1. **FUTURE SCOPE**

Sharing long, complex URLs online is not ideal – hence the need for link shorteners. However, while they have many benefits, they’re not a totally perfect solution. Unfortunately, it can be difficult to ascertain when and where it’s best to use a URL shortener, and when it’s smarter to reconsider your strategy.

The crux of it all is this: The simplest and most effective way to avoid the pitfalls is to understand how link shorteners operate. Through this knowledge, you can seek to understand how shortened URLs behave in different situations, enabling you to harness the benefits for your own gain.

URL shortening services simply make hyperlink text look more attractive. Normally, website URLs take up a bunch of space. They’re often unattractive strings of random letters and numbers, and can be distracting and aesthetically unpleasing.

However, it’s more than just aesthetics. Shortened links make the reading experience smoother, particularly on social media platforms. In addition, they make it easier for your viewers to share content, as it’s less likely they’ll make an error when copying over links (although this is less of a concern when using modern smartphones). Not only that, but there’s [no danger](http://www.oncrawl.com/url-shorteners-seo/) of losing any hard-won SEO benefits – Google are smart enough to pass on link juice for any 301 or permanent redirects implemented.

There’s also click tracking to consider, and measuring the success of your marketing efforts is always imperative. In a nutshell, you can’t improve on targeting your audience if you don’t know how they respond to your content. Link shorteners provide analytics to show you who has clicked your links

**REFERENCE**

1. <https://en.wikipedia.org/wiki/Flask_(web_framework)>
2. <https://github.com/jerrysingh181/URL-SHORTENING>
3. <https://www.upgrad.com/>
4. <https://www.fullstackpython.com/flask.html>
5. <https://realpython.com/tutorials/flask/>