Introduction in Python frameworks for web development

Introducere în cadrele Python pentru dezvoltare web

Flaviu FUIOR

Institutul Național de Cercetare-Dezvoltare în Informatică – ICI București

flaviu.fuior@ici.ro

Rezumat: Python este un limbaj de programare foarte popular care acoperă cu succes domeniul dezvoltării web. Această lucrare oferă o imagine de ansamblu asupra abordării Python pentru dezvoltarea web, prezentând diferite tipuri de cadre Python și oferind îndrumări pentru alegerea celui adecvat. În plus, sunt prezentate rezultatele pentru popularitate bazate pe indicele PYPL Popularity of Programming Language și principalele domenii în care Python este utilizat pe baza sondajului anual JetBrains. După descrierea celor mai utilizate opt cadre Python, atât full-stack cât și cele minimaliste, lucrarea se concentrează pe cele mai populare două Django și Flask, comparându-le pe baza caracteristicilor definitorii și evidențiind avantajele și dezavantajele pentru ambele.

Cuvinte cheie: Python, dezvoltare web, cadru full-stack, cadru minimalist, Django, Flask.

Abstract: Python is a very popular programming language that covers with success the web development domain. This paper provides an overview of the Python approach for web development by presenting different types of Python frameworks and offering guidance on how to choose the appropriate one. Furthermore, it presents the popularity results based on the PYPL PopularitY of Programming Language index and the main domains where Python is used based on JetBrains yearly survey. After the description of the most utilized eight Python frameworks, both full-stack and minimalistic ones, the paper focuses on the two most popular ones Django and Flask comparing them based on definitory characteristics and highlighting their respective advantages and disadvantages.

Keywords: Python, web development, full-stack framework, minimalistic framework, Django, Flask.

1. Introduction

The world of programming languages is a very dynamic one and it is constantly shifting and evolving in directions more or less expected. Sometimes is not easy to properly identify the most appropriate and in-vogue programming language dedicated to a specific development domain.

One of the most mature development domains but still very actual, dynamic, and attractive is the web development domain. On the other hand, one of the easy to learn and very popular programming language is Python.

The purpose of this article is to provide an overview of the Python approach for web development by presenting different types of Python frameworks and offering guidance on how to choose the appropriate one.

The second section explores why Python represents a good programming language choice. Furthermore, we will look at the popularity results based on the PYPL PopularitY of Programming Language index and the main domains where Python is used. In the third section there are identified and described the most used Python frameworks for web development regardless of being full-stack or minimalistic.

The fourth section presents a comparison of the most popular two Python frameworks for web development (Django and Flask) based on definitory characteristics, followed by acknowledging their advantages and disadvantages. The last section presents conclusions about Python, Django, and Flask.

2. Python programming language

Guido van Rossum, a 33 years old Dutch programmer at that time, started implementing an interpreter for a new scripting language in December 1989, and the result was Python's first release in 1991. Python is a high-level and general-purpose scripting language and also is an interpreted and a dynamically typed language. It supports numerous programming paradigms including object-oriented, functional programming, and structured (particularly, procedural). (Micheelsen, 2016)

One of the main reasons for Python's current popularity is its readability. Capability to have a readable code regardless of the author is very important for software engineers and Python step up this by being a very readable type programming language. (Ghimire, 2020) Python language is one of the most accessible programming languages available because it is not complicated and has simplified syntax and this makes it very easy to learn and use by beginners.

Python language popularity can be easily followed using the PopularitY of Programming Language (PYPL) index which is generated by studying how often programming language tutorials are searched using Google. The PYPL index is a leading indicator and uses raw data from Google Trends. The result for this index in November 2020 compared to the index values one year back, worldwide is presented in Figure 1. (PYPL, 2020)

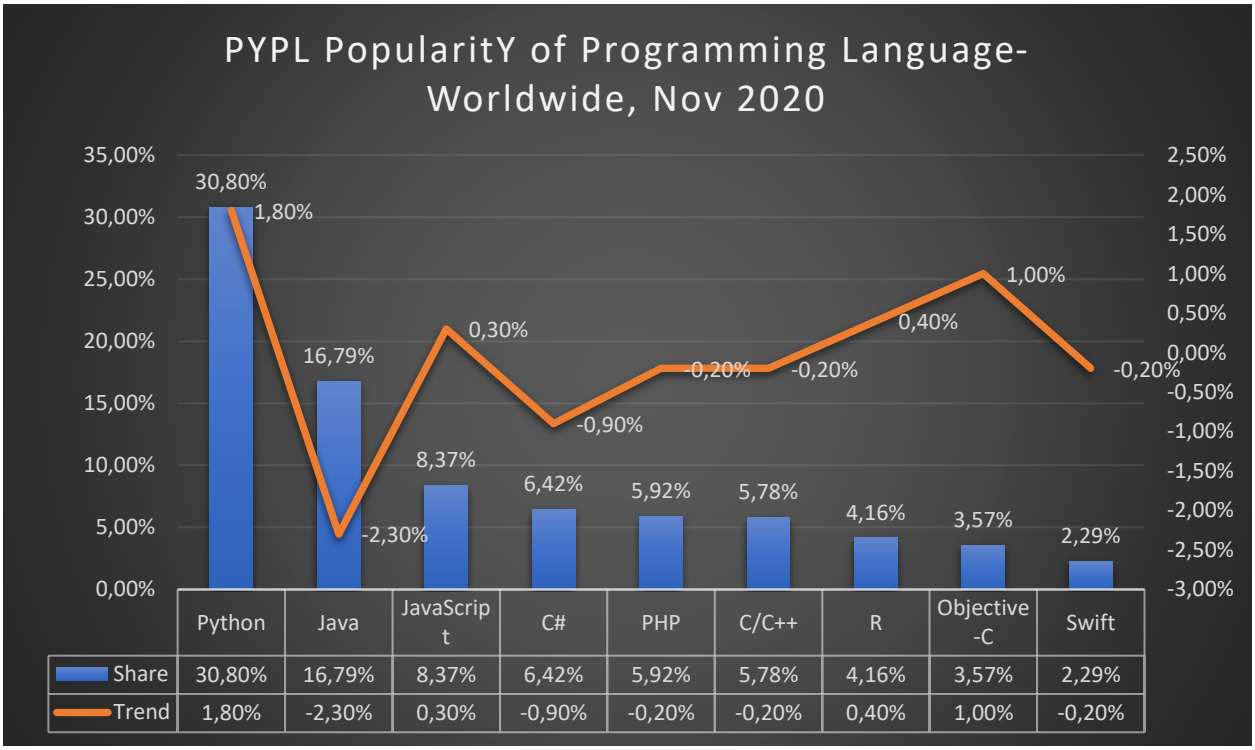

Figure 1 - PYPL PopularitY of Programming Language index - Worldwide, Nov 2020 compared to one year back (PYPL, 2020)

To be sure that Python programming language popularity is not influenced by seasonal factors, we have checked the interest for Python worldwide against the other two top languages (Java and JavaScript) using Google trends in the last three years as shown in Figure 2 below. It shows that starting with June 2019 interest for Python has surpassed interest for Java.

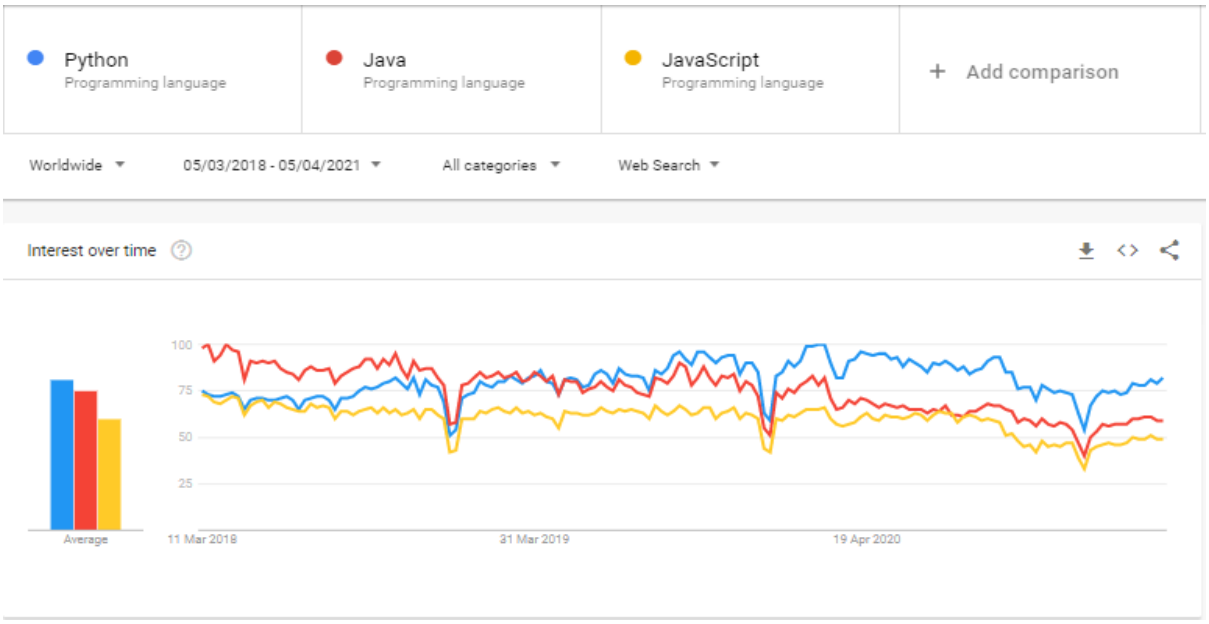

Figure 2 – Python, Java, and JavaScript interest over time worldwide (Google trends, 2021)

Python has an object-oriented architecture and is currently used not only for scripting but in many other different areas like web development, machine learning, data analysis, software testing. (Fuior, 2019) (Teodorescu, 2019) JetBrains, a software development company that offers a large family of integrated development environments (IDE), conducts a yearly survey the 3-year results based on the answers to the question "What do developers use Python for?" are presented in Figure 3.

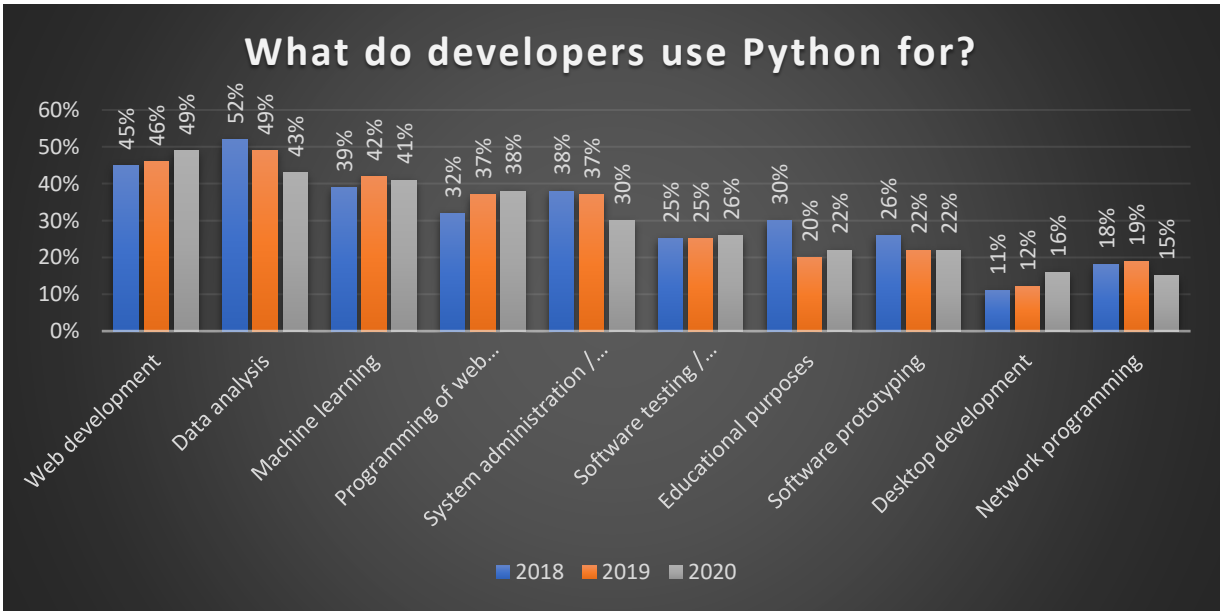

Figure 3 - What do developers use Python for? (JetBrains, 2020)

Analyzing Figure 3, it is clear that web development together with data analysis, machine learning, programming of web parsers / scrapers / crawlers, system administration / writing automation scripts / infrastructure configuration (DevOps) are the main areas where the versatile Python language is most used. Furthermore, web development with Python had an ascending trend in the last three years which led to becoming the first domain of use for Python in 2020. The main reason why Python became so popular in web development is its frameworks.

3. Python frameworks for web development

A web framework is a collection of modules or bundles that enable programmers to create web applications or services without worrying about low-level specifics like protocols, sockets, or process/thread management. The vast majority of Python's web frameworks are only server-side technology, although, with the increased prevalence of AJAX, some of Python's web frameworks are starting to include AJAX code that helps programmers with the particularly difficult task of programming (client-side) the user's browser. (Wiki.Python, 2020)

A programmer using a framework is usually writing code that conforms to some kind of protocols that lets him "plug in" to the framework, delegating duties such as the communications, infrastructure, and low-level things to the framework while focusing his code on the application logic.

There are two types of web frameworks: the minimalistic type and all-inclusive type (also called full-stack). Frameworks which are full-stack offer the developer everything he needs to start working immediately, like a template engine, some way to access and save data in databases and many more features. Other web frameworks go the minimalistic type trying to be as flexible as possible leaving the user the possibility to choose what’s best for him. (van Rossum, 2010)

The most popular Python full-stack frameworks are:

Django

Django was introduced in the autumn of 2003 in a press agency in Lawrence, Kansas when web programmers Adrian Holovaty and Simon Willison started using Python language to build applications. (Dauzon, 2016) The main motivation for developing Django was the need to develop applications faster in a matter of days or even hours.

Django is an open-source web framework for perfectionists with tight deadlines that follows the MTV (model – template - views) architectural pattern and is maintained by an American independent organization called Django Software Foundation (DSF).

The framework's main purpose is to simplify the development of complex, database-driven websites. Django emphasizes "pluggability" and reusability of elements, low coupling, shorter code, fast development, and the DRY principle (Don't Repeat Yourself). Also, this framework offers an optional administrative interface for operations like create, update, read, and delete. This interface is created dynamically via introspection and configured in admin models.

Django's key features are:

- ✓ Authentication backends;
- ✓ URL routing;
- ✓ Its own web server;

- ✓ Template engine;
- ✓ An Object-Relational Mapper (ORM);
- ✓ Middleware support;
- ✓ Database schema migrations;
- ✓ A Python unit test framework. (Tamrakar, 2019)

Web2py

Web2py is an open-source and versatile full-stack web framework for Python programmers searching for a very scalable framework. The framework has its very own web-based IDE, which, besides numerous elements, contains a code editor, debugger tool, and one-click deployment.

Web2py's ticketing tool is one of the most major features of this framework. Every time an error occurs, the tool issues a ticket to the user, and this way, the user can follow errors and their statuses. (Singh, 2020)

Web2py's key features are:

- ✓ No requirements for installation and configuration;
- ✓ Capability to function on any given web hosting platform with the condition to provide support for either Java and Python or Python;
- ✓ Follows MVC (model – view - controller) pattern;
- ✓ Readability of multiple protocols;
- ✓ Role-based access control;
- ✓ Backward compatibility;
- ✓ Data security is built-in allowing to prevent some common vulnerabilities like injection flaws, malicious file execution, and cross-site scripting;
- ✓ Offers support for internationalization.

Tornado

Tornado is one of the older Python web frameworks which centers more on the speed and network while including all the web frameworks features as well. It is mainly based on a FriendFeed framework, which was bought by Facebook in 2009, and was developed by Bret Taylor and designed to serve a huge amount of traffic without any performance downtime. (Wiki.Python, 2020)

The framework is best used when the application needs long polling, web sockets or long-lived connections. Due to its asynchronous interactive connection feature, Tornado benefits greatly from the network-intensive applications. (Ghimire, 2020)

Tornado is an open-source web framework and its key features are:

- ✓ Web templating;
- ✓ Delivers high-quality output;
- ✓ Real-time services;
- ✓ User authentication built-in support;
- ✓ Enable implementation of vendors authentication and authorization schemes;
- ✓ Non-blocking HTTP client;
- ✓ Support for translation and localization. (Singh, 2020)

The most popular Python minimalistic frameworks are:

Flask

Flask is a popular solution for both web applications and microservices and it was originally inspired by the Sinatra Ruby framework. This "micro-framework" was designed by Armin Ronacher as an extensible framework; Flask ensures a strong core with basic services, while extensions provide the rest.

This framework has three important dependencies and these are: Werkzeug WSGI (Web Server Gateway Interface) toolset; Jinja2 template and Click command-line interface. Flask does not offer native support for authenticating users, accessing databases or validating web forms. These features and other more are available via extensions that integrate with the core packages. (Grinberg, 2018)

Flask's key features are:

- ✓ Built-in fast debugger and a development server;
- ✓ Jinja2 templating;
- ✓ Integrated support for unit testing;
- ✓ Capability to plug in any ORM (Object-relational mapper);
- ✓ HTTP request handling;
- ✓ RESTful request dispatching;
- ✓ Supports secure cookies for client-side sessions;
- ✓ WSGI 1.0 compliance;
- ✓ Unicode-based. (Singh, 2020)

Bottle

Micro-framework Bottle also termed as a true Python framework is a single source file framework and was initially developed for building APIs. Bottle framework is often used for small applications and prototyping and has no dependency outside the Python Standard Library and it has less features, than other frameworks.

Bottle's key features are:

- ✓ Supplies its own templating engine called Simple Template Engine, which can be changed with Cheetah, Jinja2, or Mako;
- ✓ Built-in development web server, which can be effortlessly replaced;
- ✓ URL routing;
- ✓ Support for other WSGI servers;
- ✓ Plugin support for many databases;
- ✓ Assistance for headers, forms, cookies, and file uploads. (Ghimire, 2020)

Pyramid

Pyramid is a minimalistic open-source web framework built as a part of the Pylons Project. The central characteristic of Pyramid is its capacity to work efficiently with all sizes of web applications and its main purposeis to do as much work as feasible keeping the smallest complexity. (Tamrakar, 2019)

The framework has an MVC architecture and implementation details are similar to Flask. Pyramid does not come with any ORM but instead, it gives more flexibility and power when it comes to adding new features.

Pyramid's key features are:

- ✓ Templating system;
- ✓ Debugger mode;
- ✓ Static assets;
- ✓ Generate dynamic URLs;
- ✓ URL routing;
- ✓ Built-in support for HTTP session;
- ✓ Extendibility with add-ons (libraries).

CherryPy

CherryPy is the first framework of Python programming language released in June 2002 by Remi Delon. This micro-framework is extensible with its flexibility by design and has built-in tools such as authorization, sessions, routing, and support for databases. (Ghimire, 2020)

A major difference between CherryPy and Flask or Bottle is that CherryPy is object-oriented and concentrates on being as "pythonic" as possible. In other words, CherryPy intends to make writing a web application as similar to writing general python code as possible.

CherryPy's key features are:

- ✓ It can run easily on multiple HTTP servers simultaneously;
- ✓ Powerful configuration system;
- ✓ An adaptable inbuilt plugin system;
- ✓ HTTP/1.1-compliant WSGI thread-pooled web server;
- ✓ Built-in assistance for coverage, profiling, and testing.

Falcon

Falcon is a performance-focused micro-framework for building REST APIs and microservices. The framework is named like the falcon, representing how fast Falcon operates.

This micro-framework allows programmers to handle most requests and craft cleaner designs and Falcon never opposes its programmers in the selection of libraries for authorization and databases. (Tamrakar, 2019)

Falcon's key features are:

- ✓ An extendible, highly-optimized codebase;
- ✓ Natural HTTP error responses;
- ✓ DRY demand processing via middleware elements and hooks;
- ✓ A simple approach for headers and bodies by using request and response classes;
- ✓ REST-inspired resource classes and URI templates allow intuitional routing;
- ✓ Upfront exception handling;
- ✓ Unit testing using WSGI helpers and mocks. (Singh, 2020)

Considering the above frameworks, it is important to have an overview of how much popularity each full-stack or minimalistic framework has in order to identify the most appreciated and used frameworks. According to JetBrain's yearly survey, the results recorded in the last two years for the question "What web frameworks or libraries do developers use in addition to Python?" are presented below in Figure 4.

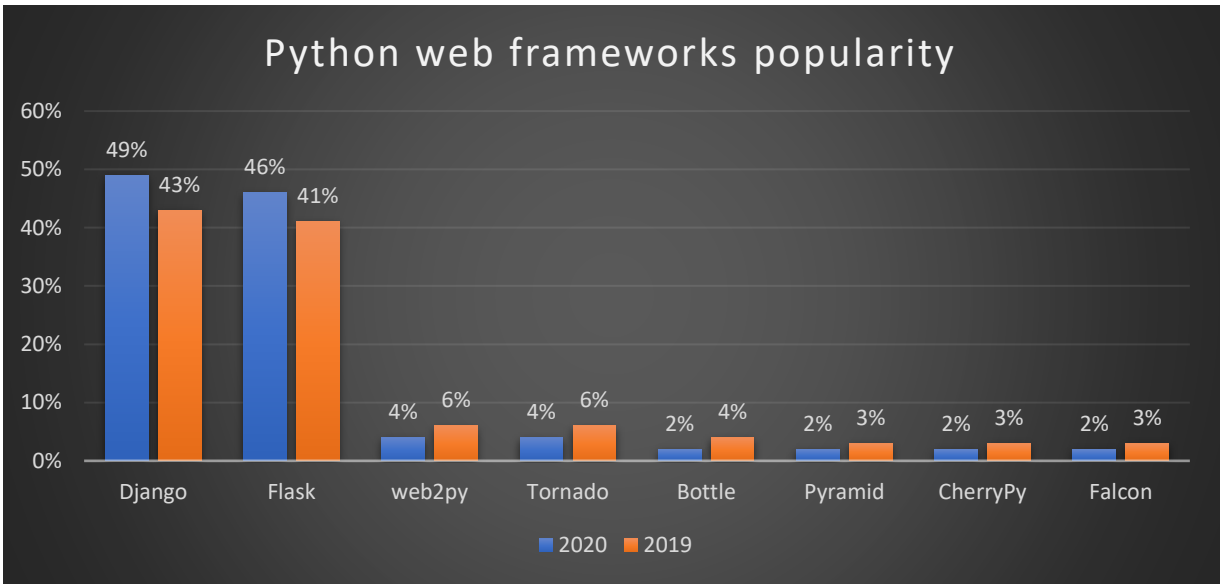

Figure 4 - What web frameworks or libraries do developers use in addition to Python? (JetBrains, 2020)

Analyzing the figure above, it is clear that the most used Python web frameworks are the full-stack Django and the minimalistic Flask, while the rest of the frameworks remain far behind in terms of popularity. This is the reason why in the next section we will further compare these two frameworks.

4.Django vs Flask

This section will present a comprehensive comparison of the most popular Python web frameworks: Django and Flask, based on 19 definitory characteristics as summarized in Table I below.

Characteristics	Django	Flask
Framework type	Django is a full-stack web framework based on the batteries-included approach.	Flask is a lightweight (micro) framework with minimalistic characteristics.
Ready-to-use Option	It has an embedded Django admin framework that can be quickly customized.	It doesn't offer any ready-to-use option to manage admin tasks.
Controlling	Developers have access from the beginning to the most usual characteristics that make development simpler and faster.	Developers can learn and maintain control of the application core.

Template Engine +	It has natively an inbuilt template system and this helps to save development time.	Its template system, named Jinja2, is based on Django's template system.
Admin Tool	Django's administration tool is an inbuilt bootstrapping instrument with which programmers are able to create web applications without any outside input.	Administration characteristics are not as obvious as in the Django web framework.
Databases	Django doesn't accept several kinds of databases but it accepts SQLite, MySQL, PostgreSQL.	Flask permits any database including NoSQL.
Execute Database Tasks	With an inbuilt ORM tool, Django permits programmers to utilize almost whatever database and execute usual database tasks without writing any large queries.	In Flask programmers have to operate in various databases utilizing ORM schemes for Python and SQLAlchemy as a SQL toolkit. SQL queries are required for usual tasks.
Visual Debug	Django doesn't support Visual Debug.	Flask supports Visual Debug.
Single Application	Django enables developers to break a unique project into many smaller applications which makes them simple to build and maintain.	A project can be a unique application, although many models and views can be attached to a single application.
Production-ready Framework	Django is well known as a production-ready framework.	Flask is single-threaded and it is possible not to perform so well when put under heavy pressure.
Working Style	Django presents a uniform working style.	It offers a diversified working style.
Variable	All the views are put as a separate parameter.	The request-oriented object is imported out of the Flask module, which is an extensive variable in Flask.
Popularity	Django is perceived to be a little bit more popular than Flask because it gives numerous interesting characteristics and decreases the time needed to develop complex applications.	Flask is a very helpful framework if programmers have recently started with web development.
Flexibility	The developers don't have substantial flexibility because of the modules given by Django.	The programmers are totally free to utilize any plugins and libraries and flexibly develop any functionalities.
Alterations	Django is not recommended for projects where the specifications modify dynamically.	With Flask, a simple application can be later modified to supply extra functionality and build it multifaceted. Flask offers the flexibility to extend the application fast.
Third-party Applications	The framework supports a huge quantity of third-party applications.	The framework does not provide support for third-party applications.
API support	It doesn't offer support for API.	It has support for API.

Lines of Code	For identical functionality, Django requires more than double more code lines than the Flask web framework.	Flask web framework requires fewer code lines for any simple task.
URL Dispatcher	The URL dispatcher of Django is based on controller-regex.	The URL dispatcher of Flask is a RESTful request.

Table I - Django vs Flask - comparison of the most popular Python web frameworks (Mistry, 2020)

To complete the comparison between Django and Flask, the paper proposes a final approach from a different angle that will put side by side advantages and disadvantages for both these two popular Python web frameworks. Table II, below, present this pros and cons approach.

	Pros	**Cons**
Django	Offers an easy to utilize interface for more administrative events	Requires a large size of code
	Easier to set-up and execute	It's a monolithic platform
	Provides multilingual websites using its inbuilt internationalization scheme	Big reliance on Django ORM. Wide knowledge is required
	REST framework has much more support for numerous authentication protocols	Offers an advanced entry point even for simple resolutions
	Used for rate-limiting API request of a single user	Much enlarged for small schemes
	Permits to document API with HTML output	Has a high learning curve
	Provides an inbuilt authentication arrangement	Permits to handle only a single request at a given moment
	Assists developers to set patterns for URLs in the application	Auto-reload restarts the entire server
	It's a high-level framework for quick web development	Developers can organize components collectively; this can arise confusion
	Provides an entire stack of tools	Routing demands a certain knowledge of usual expressions
	Pros	**Cons**
Flask	Easier to utilize for simple situations	Setting-up any big project needs some knowledge of this web framework
	Has great scalability for simple applications	High maintenance fees for more multifaceted systems
	Code size is moderately smaller	Complex maintenance for bigger applications implementations
	Database integration is simpler and intuitive	Supplies less assistance and has a smaller community as opposed to Django framework

Routing URL is easy	Slow MVP (minimum viable product) development in most situations
Simple to develop a rapid prototype	Async can be an issue with this framework
Easy to build and maintain applications	Absence of ORM and database
Many resources accessible online, especially on GitHub	
A small core and effortlessly extensible	
Minimal, yet authoritative platform	

Table II - Django and Flask - advantages and disadvantages (Mistry, 2020)

Finally, it is worth mention what big companies are using Django and what big companies are using Flask as in Table III below.

Who is using Django?		**Who is using Flask?**	
• **YouTube** • **Instagram** • **Spotify** • **DropBox** • **The Washington Post** • **National Geographic**	• **NASA** • **Pinterest** • **Bitbucket** • **Eventbrite** • **Quora** • **Udemy**	• **Red Hat** • **Airbnb** • **Netflix** • **Reddit** • **MIT** • **Samsung**	• **Rackspace** • **Lyft** • **Mailgun** • **Teradata** • **Rackspace** • **Hotjar**

Table III - Who is using Django and Flask? (Geeksforgeeks, 2020) (Github, 2021)

5. Conclusions

It has become a certitude that Python is the most popular programming language and has an ascending trend. One of the main reasons for its popularity is Python's versatility which makes it the choice for developers for various tasks like machine learning, web development or even software testing.

An interesting fact is that even if Python is used in many different areas, in the last years, web development with Python has become more and more interesting for developers, reaching its peak in 2020 when it was chosen as the first domain of use for Python.

Python frameworks for web development are considered to be the reason behind Python's popularity in this domain, and the greater variety of choices between full-stack frameworks (Django, Web2py, Tornado) and minimalistic frameworks (Flask, Bottle, Pyramid, CherryPy, Falcon) confirm the reason. Between numerously Python web frameworks, two of them, the full-stack Django framework and the minimalist Flask framework, distinguish themselves as the most popular frameworks.

After a comprehensive comparison between these two frameworks based on 19 characteristics and a clear highlight presentation of pros and cons for Django and Flask, the paper shows what big companies

use Django and Flask. It's clear that both frameworks are very reliable but each framework has its advantages and disadvantages and the decision to use one or another framework should keep in mind various subjective factors (tight deadline, team experience, framework learning curve, and other technical aspects).

In the coming years it will be interesting to follow on one hand the Python evolution as the most popular programming language and on the other hand the evolution of Python web frameworks especially Django and Flask, and why not the rise in popularity of some new frameworks.

Acknowledgment

This work was supported by project PN 19 37 04 01"New solutions for complex problems in current ICT research fields based on modeling and optimization", funded by the Romanian Core Program of the Ministry of Research and Innovation (MCI), 2019-2022.

BIBLIOGRAPHY

1. Dauzon Samuel, Bendoraitis Aidas, Ravindran Arun (2016) - *Django: Web Development with Python* August, 2016 – Packt Publishing
2. Fuior Flaviu (2019) - *An overview of some tools for automated testing of software applications* - Romanian Journal of Information Technology and Automatic Control, Vol. 29, No. 3, 97-106, 2019
3. Geeksforgeeks (2020) - *Top 10 Django Apps and Why Companies Are Using it?* -April 18, 2020 - https://www.geeksforgeeks.org/top-10-django-apps-and-why-companies-are-using-it/
4. Ghimire Devndra (2020) - *Comparative study on Python web frameworks: Flask and Django* – May 5, 2020 - Metropolia University of Applied Sciences
5. Github (2021) - *Companies using Flask* - *https://github.com/rochacbruno/flask-powered*
6. Google trends (2021) - https://trends.google.com/trends/explore?date=2018-03-05%202021-04-05&q=%2Fm%2F05z1_,%2Fm%2F07sbkfb,%2Fm%2F02p97
7. Grinberg Miguel (2018) - *Flask Web Development: Developing Web Applications with Python* – March 02, 2018 -O'Reilly Media, Inc
8. JetBrains (2020)- *Python Developers Survey 2020,2019, 2018* - available at: https://www.jetbrains.com/lp/devecosystem-2020/python/ https://www.jetbrains.com/lp/devecosystem-2019/python/ https://www.jetbrains.com/research/devecosystem-2018/python/
9. Micheelsen Stefan, Thalmann Bruno (2016) - *PyT - A Static Analysis Tool for Detecting Security Vulnerabilities in Python Web Applications* – May 31, 2016 - Aalborg University Computer Science
10. Mistry Jigar (2020) - *Flask vs Django: A Detailed Comparison of Python Web Frameworks* – November 27, 2020 - available at: https://www.monocubed.com/flask-vs-django/
11. PYPL (2020) - *PYPL PopularitY of Programming Language* - available at: http://pypl.github.io/PYPL.html

12. Singh Vijay (2020) - *Best Python Frameworks* – August 6, 2020 - available at: https://hackr.io/blog/python-frameworks
13. Tamrakar Anand, Daga Meenakshi, Lunia Anusha (2019) - *Illustration on the Superiority of Django Over the Other Python Web Frameworks* – December, 2019- Journal of The Gujarat Research Society -Volume 21
14. Teodorescu Paul (2019) - *Recunoașterea unei cifre scrise de mână folosind o rețea neuronală convoluțională și biblioteca TensorFlow* - Romanian Journal of Information Technology and Automatic Control, Vol. 29, No. 4, 47-62, 2019
15. van Rossum Guido, Drake Fred L. Jr. (2010) - *HOWTO Use Python in the web - Release 2.6.4* - January 04, 2010 - Python Software Foundation
16. Wiki.Python (2020) – *Web Frameworks* available at: https://wiki.python.org/moin/WebFrameworks

Flaviu FUIOR a absolvit Facultatea de Finanțe, Asigurări, Bănci și Burse de Valori din cadrul Academiei de Studii Economice București în anul 2003. În 2005 a terminat un master în management la Școala Națională de Studii Politice și Administrative București. Are peste 15 ani de activitate în organizații private și de stat în domeniul TIC. Domeniile sale principale de interes sunt: inteligența artificială, securitate cibernetică, internetul obiectelor (IoT), testare automată, protecția infrastructurilor critice, metodologii pentru managementul proiectelor.

Flaviu FUIOR graduated the Faculty of Finance, Insurance, Banking and Stock Exchanges of the Bucharest Academy of Economic Studies in 2003. In 2005 he completed a Master in Management at the National School of Political and Administrative Studies Bucharest. He has over 15 years of activity in private and state organizations in the IT&C field. His main areas of interest are Artificial Intelligence, cybersecurity, Internet of Things, automated testing, critical infrastructure protection, project management methodologies.