СЪДЪРЖАНИЕ:

[1 Увод 3](#_Toc133503196)

[2 Работа с файлове 3](#_Toc133503197)

[3 Python файлове 4](#_Toc133503198)

[4 За какво се използва Python и какви са ползите му? 6](#_Toc133503199)

[5 Какво е Python? 7](#_Toc133503200)

[5.1 Python - Стандартни библиотеки 7](#_Toc133503201)

[6 Операционна система (ОС) 9](#_Toc133503202)

[6.1 Основни функции на операционната система 11](#_Toc133503203)

[6.2 Управление на процесора 12](#_Toc133503204)

[6.3 Управление на паметта 12](#_Toc133503205)

[6.4 Потребителски интерфейс 13](#_Toc133503206)

[7 Ftp 13](#_Toc133503207)

[8 Request 14](#_Toc133503208)

[9 Django REST Framework 20](#_Toc133503209)

[10 Django 21](#_Toc133503210)

[11 Как сами да създавате и импортирате модули? 22](#_Toc133503211)

[11.1 Създаване и импортиране на модули 22](#_Toc133503212)

[11.2 Как се пише модул? 22](#_Toc133503213)

[11.3 Достъп до модули от друга папка 24](#_Toc133503214)

[12 Какво е HTTP? 25](#_Toc133503215)

[12.1 Уеб браузърът и уеб сървърът си говорят на един език – HTTP 26](#_Toc133503216)

[13 Библиотеки на Python 27](#_Toc133503217)

[13.1 Pandas 27](#_Toc133503218)

[13.2 NumPy (Numerical Python) 28](#_Toc133503219)

[13.3 Python Matplotlib 29](#_Toc133503220)

[13.4 Seaborn 30](#_Toc133503221)

[13.5 Scikit-learn 31](#_Toc133503222)

[13.6 XGBoost 33](#_Toc133503223)

[13.7 TensorFlow 34](#_Toc133503224)

[13.8 Keras 35](#_Toc133503225)

[13.9 PyTorch 36](#_Toc133503226)

[13.10 NLTK (Natural Language Toolkit) 37](#_Toc133503227)

[14 Обяснение на кода 38](#_Toc133503228)

[15 Обяснение на клиентския код 40](#_Toc133503229)

[16 Списък на използваната литература 42](#_Toc133503230)

# Увод

Проектът се дели на три модула. Клиентски модул, сървърен модул и конфигурационен модул.С моите съученици работихме заедно през последните 5 месеца. В свободното си време се разбирахме и се събирахме на различни места ,за да направим кода си. Работихме здраво и усърдно ,за да стане всичко както трябва, с една дума идеално. Имахме много спънки ,но не се отказахме ,не бяхме сами Господин Бориков ни помагаше много ,когато се затруднявахме. Моята част е да направя FTP клиент които да се интегрира лесно и да се свързва автоматично със сървъра и да започне да качва файлове на него които подлежат на архивиране. Другата половина от проекта е работа с Операционната система и слава богу има разработени достатъчно библиотеки. По стандартната част на проекта е да се претърси конкретната папка за конкретен вид файлове ,да се извадят атрибутите ,да се свери дали файла е модифициран след последното архивиране и ако е да се архивира.

# Работа с файлове

Работата с файлове е важна част от всяко уеб приложение.

Python има няколко функции за създаване, четене, актуализиране и изтриване на файлове. Ключовата функция за работа с файлове в Python е функцията open(). Функцията open() приема два параметъра; име на файл и режим. Има четири различни метода (режима) за отваряне на файл:

"r" - Четене - Стойност по подразбиране.

Отваря файл за четене, грешка, ако файлът не съществува

"a" - Добавяне - Отваря файл за добавяне, създава файла, ако не съществува

"w" - Записване - Отваря файл за запис, създава файла, ако не съществува

"x" - Създаване - Създава посочения файл, връща грешка, ако файлът съществува

Освен това можете да посочите дали файлът да се обработва като двоичен или текстов режим

"t" - Текст - Стойност по подразбиране. Текстов режим

"b" - Двоичен - Двоичен режим (напр. изображения)

Синтаксис:

За да отворите файл за четене е достатъчно да посочите името на файла:

f = отворен ("demofile.txt")

Кодът по-горе е същият като:

f = отворен ("demofile.txt", "rt")

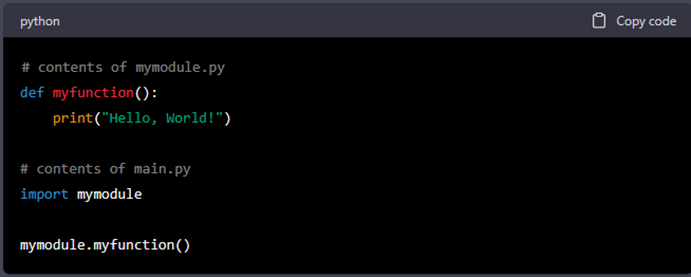
Тъй като "r" за четене и "t" за текст са стойностите по подразбиране, не е необходимо да ги указвате.

# Python файлове

В Python файлът е колекция от данни, които се съхраняват на диск или друго устройство за съхранение. Python файловете могат да се използват за съхраняване на код на Python или всеки друг тип данни като текст, изображения, аудио и видео файлове. Файловете с код на Python обикновено имат разширение и съдържат код на Python, който може да бъде изпълнява се от интерпретатор на Python. Файловете с код на Python могат да се създават и редактират с помощта на произволен текст редактор или интегрирана среда за разработка (IDE) като Visual Studio Code, PyCharm или Spyder. Когато стартирате файл на Python, интерпретаторът на Python чете и изпълнява кода във файла от горе до долу. Можете да стартирате Python файл от командния ред, като напишете



Файловете на Python могат също да импортират код от други файлове на Python, като използват инструкцията за импортиране. Това ни позволява да организираме кода си в модули и да използваме повторно код в множество файлове. Ето един пример за това как да импортираме Python модул от друг файл:



В този пример имаме два файла: "mymodule.py" и "main.py". Файлът "mymodule.py" съдържа функция, наречена "myfunction()", а файлът "main.py" импортира модула "mymodule", използвайки оператора "import". И накрая, файлът "main.py" извиква функцията "myfunction()" от модула "mymodule", използвайки нотацията с точка.

Python файловете могат също да се използват за съхраняване и манипулиране на данни. Python предоставя няколко вградени модула за работа с различни файлови формати като CSV, JSON и XML. Можем също да създадете свои собствени файлови формати, като използваме вградените I/O функции на Python.

Python е популярен език за програмиране на високо ниво, който има голяма и активна общност от разработчици. Има много налични библиотеки за Python, които предоставят допълнителна функционалност и инструменти за различни задачи. Някои от най-популярните библиотеки в Python включват:

1.NumPy: NumPy е мощна библиотека за научни изчисления в Python. Той осигурява поддръжка за големи, многомерни масиви и матрици, както и функции за математически операции върху тези масиви.

2.Pandas: Pandas е библиотека за анализ на данни и манипулиране в Python. Той предоставя структури от данни и функции за работа с таблични данни, като CSV файлове и електронни таблици на Excel.

3.Matplotlib: Matplotlib е библиотека за създаване на визуализации и графики в Python. Предоставя разнообразие от функции за създаване на линейни диаграми, точкови диаграми, хистограми и др.

4.Scikit-learn: Scikit-learn е библиотека за машинно обучение в Python. Той предоставя набор от инструменти за предварителна обработка на данни, избор на функции и обучение и оценка на модели.

5.TensorFlow: TensorFlow е библиотека с отворен код за машинно обучение, разработена от Google. Той предоставя инструменти за изграждане и обучение на невронни мрежи, както и поддръжка за други алгоритми за машинно обучение.

6.Flask: Flask е лека уеб рамка за Python. Той предоставя инструменти за изграждане на уеб приложения и API, както и поддръжка за шаблони и маршрутизиране.

7.Pygame: Pygame е библиотека за изграждане на игри и мултимедийни приложения в Python. Предоставя инструменти за работа с графики, звук и въвеждане от потребителя.

Има много други налични библиотеки за Python, покриващи широк спектър от теми от обработка на естествен език до уеб скрапинг до криптография. Обширната библиотечна екосистема на Python е една от ключовите му силни страни, което го прави мощен инструмент за широк набор от задачи.

# За какво се използва Python и какви са ползите му?

[Python](https://www.python.org/) е един от най-популярните програмни езици, който се използва в много и различни сфери – от разработката на уеб приложения и такива за десктоп среда, до анализ на данни и управление на обучителни модели за [машинно обучение](https://bg.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE_%D1%81%D0%B0%D0%BC%D0%BE%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) (machine learning).

# Какво е Python?

[Python](https://help.superhosting.bg/python.html) (произнася се „пайтън“) е език за програмиране, създаден от [Гуидо ван Росум](https://en.wikipedia.org/wiki/Guido_van_Rossum" \t "_blank) през 1990 г. Python e програмен език с общо предназначение, за разлика например от PHP, чието основно предназначение е за използване в уеб. Python е много по-лесен за научаване, сравнен с други програмни езици. Приложенията, написани на Python, са компактни и с лесно четим код за разработчиците, като често са и по-кратки от еквивалентните им, написани например на C/C++. Python е инсталиран в повечето GNU/Linux дистрибуции. Създателят на Python е фен на британската комедийната група „[Monty Python](https://en.wikipedia.org/wiki/Monty_Python)“. И в нейна чест той кръщава програмния си език Python. Разработчиците, които използват Python, поставят на първо място следните му ползи: страхотни библиотеки, голямо подкрепящо общество и фенове, четим код, лесна и бърза разработка. Една от другите големи ползи на езика, още преди разработчикът да е започнал да го използва, е, че е лесен за научаване. Това го прави много достъпен за начинаещи програмисти.

## Python - Стандартни библиотеки

Едно от нещата, които е по-привлекателно да се развиват в **Python,** е неговата богата и страхотна библиотека от стандартни модули, тя има много помощни програми, до които можем да имаме достъп, като само правим импорт на модулите, които покриват нашите нужди.

**SYS модул.** Този модул ни дава достъп до променливи и функции, които са тясно свързани с интерпретатора **Python**, след това ще изброим онези, които считаме за най-важни, но можете да продължите да проучвате и да задавате свои собствени предпочитани.

**argv:** Аргументите на командния ред, включително името на скрипта.**изход ([arg]):** Излиза от текущата програма, по желание с аргумент можем да предадем стойност, която ни дава връщане или съобщение за грешка.**модули:** Това е речник, който ни прави карта с имената на всички заредени модули при извършване на повикване на функцията.

**path:** Прави списък, където можем да видим местата, където можем да намерим наличните модули.

**платформа:** Съдържа идентификатор на платформа като **sunos5** или **win32.**

**stdin:** Това е стандартният входен поток, това е файл тип обект.

**stdout:** Това е стандартният изходен поток, това е файл тип обект.

**stderr stdout:** Това е стандартният поток от грешки, това е файл от тип обект.

Нека да разгледаме малък пример за това как да използваме една от тези функции, в този конкретен случай ще използваме argv, първо ще поставим кода и след това обяснението:

# reverseargs.py

import sys

args = sys.argv [1:]

args.reverse ()

print '' .join (args)

Тук можем да наблюдаваме, че това, което правим, е да вземем аргументите и след това да ги отпечатаме в обратен ред.Сега ще се обадим на нашата програма и ще видим съответния изход, който ни дава в следното изображение:

Доста впечатляващо, тъй като с няколко реда бихме могли да направим това, всичко благодарение на стандартните **Python** библиотеки.**ОС модул** Този модул ни дава достъп до множество услуги на операционната система, това е може би един от най-обширните модули, така че ще обхванем само няколко, но доста полезни от неговите функции, този модул съдържа и достатъчно функционалности за работа с директории и файлове в на системата, остава задача да се проучи официалната документация в това отношение.**среда:** Направете карта с променливите на средата на операционната система.**system** (команда): Изпълнява команда в операционната система в рамките на *подподкал* .

**sep:** Сепаратор, използван при изграждането на пътеките.

**pathsep:** Разделител, използван за разделяне на пътища.

**linesep: Разделител на** линии ('\ n', '\ r' или '\ r \ n')

**urandom** (n): Връща n байта на случайни данни със силно криптиране. Нека разгледаме един прост пример, използващ **системната** функционалност за стартиране на уеб браузър от скрипт Python. В UNIX можем да направим следното:

os.system ('/ usr / bin / firefox')

Но ние също имаме еквивалент в Windows:

os.system (r'c: \ "Program Files" \ "Mozilla Firefox" \ firefox.exe ")

В този случай забелязваме, че поставяме двойни кавички за папките на пътя, като по този начин предотвратяваме празното пространство, за да ни накара да прескочим проблем в DOS кой е този, който изпълнява тази команда с Python.

# **Операционна система (ОС**)

Операционната система (ОС) е основна част от компютърния системен софтуер, която управлява и координира ресурсите на хардуера и софтуера и обслужва изпълняваните компютърни програми. Приложният софтуер обикновено има нужда от ОС, за да работи.

ОС разпределя задачите по време, планира ефикасното използване на ресурсите на системата и може да включва специализиран софтуер за изчисление на ресурси: стойност на процесорното време, използвана памет, ресурси за печат и други. При входно-изходнитеоперации и динамично разпределение на паметта ОС действа като посредник между приложния софтуер и хардуера, макар че приложните команди обикновено се изпълняват директно, чрез системни повиквания към функции на ОС. ОС се среща на повечето устройства с компютри – от мобилни телефони през игрални конзоли до уеб сървъри и суперкомпютри. В компютърната техника операционната система (ОС) е софтуер, който управлява компютърни хардуерни и софтуерни ресурси и предоставя общи услуги за компютърни програми. Ето някои често срещани операционни системи:

1. Windows: Разработена от Microsoft, Windows е широко използвана операционна система за персонални компютри. Той предоставя графичен потребителски интерфейс и поддържа широк набор от софтуерни приложения.

2.macOS: Разработена от Apple, macOS е операционната система, използвана на Macintosh компютрите на Apple. Той осигурява удобен за потребителя интерфейс и е известен със своята стабилност и сигурност.

3.Linux: Linux е безплатна операционна система с отворен код, която се използва широко в сървъри и суперкомпютри. Той предоставя широк набор от инструменти и помощни програми за разработчици и системни администратори.

4.Android: Android е популярна операционна система за мобилни устройства, разработена от Google. Базиран е на ядрото на Linux и предоставя лесен за използване интерфейс за мобилни устройства.

5.iOS: Разработена от Apple, iOS е операционната система, използвана на iPhone и iPad. Той осигурява удобен за потребителя интерфейс и е известен със своите функции за сигурност.

6.Chrome OS: Разработена от Google, Chrome OS е лека операционна система, предназначена за използване на Chromebook. Базиран е на ядрото на Linux и предоставя удобен за потребителя интерфейс за уеб сърфиране и приложения за продуктивност.

Това са само няколко примера от многото операционни системи, които съществуват днес. Всяка операционна система има свои уникални характеристики и силни страни и изборът на операционна система зависи от специфичните нужди на потребителя.

Терминът операционна система, най-вероятно ви е познат още от ерата на първите персонални компютри. Днес, умните устройства вече са десетки видове, тяхната работа нямаше да бъде толкова продуктивна, ако не разполагаха с операционни системи, които да ни позволяват не само бързодействие на процеса, който сме им задали, но и мултитаскинг, за да използваме времето си още по-ефективно. Всъщност не всеки компютър разполага с операционна система – този, който управлява микровълновата ни фурна, например, няма нужда от такава, тъй като наборът от задачи, които трябва да изпълнява никога не се променя. От операционна система, обаче, имат нужда всички останали устройства, които изпълняват различни цели, комуникират с потребителите по по-сложен начин и могат да бъдат надграждани с цел отговаряне на променящите се нужди на хората.

Както сами можете да се досетите, нашите [смартфони](http://revo.bg/telefoni.html) и [таблети](http://revo.bg/tablets.html) отговарят на тези условия, което обяснява и защо те не могат да бъдат интелигентни мобилни устройства без да разполагат с операционна система. И макар всеки да знае какво е операционната система на практика, тук ще погледнем към нея от теоретична гледна точка.

Всяко едно устройство, което разполага с операционна система (ОС), може да промени своята работа с модифициране на същата тази операционна система. Точно поради тази причина основният софтуер на устройствата (операционната система) е съставена от така наречен „портативен” код, който търпи изменения и може да бъде надграден без промяната да окаже каквото и да е въздействие върху хардуера.

## Основни функции на операционната система

Операционната система изпълнява две основни функции, описани по-долу:

1. Тя управлява хардуерните и софтуерните ресурси на системата, към които спадат неща като: централен процесор, памет, екран, списък с контакти, батерия, свързване към интернет мрежа и какво ли още не. Тази функция е жизненоважна, тъй като многобройни програми и методи за въвеждане на информация се съревновават едновременно за „вниманието” на централния процесор и едновременно изискват оперативна памет и пространство. Целта на ОС, следователно, е да разпределя наличните ресурси между всички тези приложения по възможно най-оптимален начин.
2. ОС осигурява стабилен и постоянен начин, по който приложенията да се справят с хардуера без да знаят подробности за този хардуер. Този „начин” е по-известен като приложно-програмен интерфейс (API), който позволява на един програмист да напише приложение на едно компютърно устройство и да бъде почти 100% сигурен, че приложението ще бъде разпознато от друго устройство от същия тип.

## Управление на процесора

Управлението на процесора се изразява в две неща: гарантиране, че всеки процес или приложение получава достатъчно от времето на процесора, за да функционира и използване на колкото се може повече цикли на процесора за реална работа. В зависимост от начина, по който ОС управлява споменатите процеси различаваме два вида операционни системи: single-tasking и multi-tasking. Най-просто казано, първият вид ОС ни оставя с впечатлението, че компютърът работи само по една задача, докато вторият – че работи едновременно по много. Мобилните операционни системи, които ползваме днес, са оборудвани с отлични възможности за мултитаскинг.

## Управление на паметта

Управлението на паметта се изразява в следните две задачи: 1) на всеки процес трябва да бъде осигурена достатъчно памет, без това да пречи на останалите процеси и 2) различните видове памет в системата трябва да се използват правилно, така че всеки процес да се изпълнява с оптимален ефект.

Управлението на капацитета на твърдия диск (при мобилните устройства това е вграденият сторидж) е най-бавно. Освен с него, обаче, ОС се занимава с още два вида памет:

Високоскоростен кеш: това е бързо, сравнително малко количество памет, до което централният процесор достига най-бързо. Контролерите на кеша предвиждат от кои данни ще има нужда процесорът и след това изтеглят тези данни от основната памет в кеша, за да ускорят производителността на системата.

Основна памет: позната още като RAM памет.

Целта на ОС е да балансира между всички тези видове памет, придвижвайки данните на блокове (наричани още страници) между наличната памет в съответствие с протичащите процеси.

## Потребителски интерфейс

Когато мислим за операционни системи, ние обикновено си представяме техния „външен вид” или потребителския интерфейс. Потребителският интерфейс всъщност е нещо като „горния слой” на операционната система, а управлението на всички основни функции се случва в „ядрото”. Връзката между ядрото на операционната система и потребителския интерфейс определя и разликите между съвременните операционни системи.

# Ftp

FTP е съкращение от File Transfer Protocol и, както показва названието, представлява протокол за трансфер /прехвърляне/ на файлове. FTP е един от най-старите протоколи в интернет, но продължава да бъде най-удобния, най-простия и най-ефективния начин за прехвърляне на файлове от и към някакъв отдалечен сървър. Работата с протокола FTP изисква две отделни програмни части - сървър и клиент.

FTP-сървъра е апликация, която обикновено работи на свръхмощен компютър - с много по-голям хард диск и по-мощен процесор от тези на обикновен персонален компютър. Такива компютри се наричат сървъри и обикновено съдържат огромно количество файлове, така че на външен вид програмата FTP-сървър изглежда като голяма библиотека за файлове. FTP сървър може да се разположи и на персонална машина.

FTP-клиента (ftp-client) представлява програма, която трябва да имаме инсталирана на личния си компютър, за да сте в състояние да прехвърляте файлове от и към други отдалечени машини. Обикновено FTP клиента се използва за "качване" (upload) на сайтове в интернет, т.е. за прехвърляне на файлове от персоналния компютър към избрания хостинг.

Как да осъществя връзка с FTP сървъра на хостинга ми ? За да осъществите връзка с вашия хостинг вие трябва да имате FTP клиентска програма. В тази статия ще използваме най - разпространената и най - рационална програма - FileZilla (програмата поддържа и български език)

След като свалиме програмата и я инсталираме вие трябва да я стартираме.

За да осъществиме връзка с нашия хост, където ще поместим файловете от сайта ни, трябва да разполагаме със следните данни:

1. FTP Адрес (Хост) - това е нашия домейн, или ако не им сървърите на домейна са в процес на обновяване можете да се свържеме с IP адреса на хостинга

2. Потребител - това е нашия хостинг FTP потребител

3. Парола - паролата на FTP потребителя ни

4. Порт - 21 или просто го оставяме празно (21 е порт по подразбиране)

# Request

Модулът requests ни позволява да изпращаме HTTP заявки с помощта на Python.

HTTP заявката връща [Response Object](https://www.w3schools.com/python/ref_requests_response.asp) с всички данни за отговор (съдържание, кодиране, състояние и т.н.).

Изтеглеме и инсталирайме модула за заявки.

Навигирайме в командния ред до местоположението на PIP и въвеждаме следното:

pip install requests

[Модулът requests](https://docs-python.ru/packages/modul-requests-python/) е елегантна и проста HTTP библиотека за Python, създадена за хората. Модулът прави изключително лесно изпращането на HTTP/1.1 заявки. Няма нужда ръчно да съставяте URL адреси или да кодирате данни за PUT и POST заявки (в момента можете да използвате Request.json)

Всеки път, когато нашият браузър изтегля уеб страници, той използва метода GET за това. Това е информация за сървъра, който хоства тази страница, че съдържанието на страницата трябва да бъде върнато в нашия браузър. Да приемем, че искаме да изтеглим съдържанието на фиктивен уебсайт www.localacje-miast.com. Сайтът съдържа географски координати на градове.

Така нашият браузър изпраща запитване, подобно на следното:

https://www.likalacje-miast.com

След това сървърът, който съхранява тази страница, знае, че браузърът иска да изтегли нейното съдържание.

 Понякога сървърът очаква браузърът да предостави допълнителни параметри, които ще му позволят да посочи какво точно трябва да върне за него. Подобно на името на града, той знае кои точно координати да върне. Така нашият браузър изпраща запитване като:

 https://www.likalacje-miast.com city=Chicago

Понякога, за да изтеглите съдържанието на страницата, трябва да въведете потребителско име и парола

https://www.likalacje-miast.com city=Chicago auth=login+password

Вторият и най-популярен метод за правене на HTTP заявки е POST. Докато GET беше заявка за изтегляне на съдържание, POST се използва за изпращането му. Така че ние го използваме, когато искаме да изпратим информация до сървъра.

Имаме работа с пример за използване на POST, когато попълваме формуляри на уебсайтове. Браузърът събира въведените от нас данни и изпраща POST устройства. Услугата, виждайки POST заявката, е готова да прочете съдържащите се в нея данни и например да ги запише в базата данни. Например:

 https://formlularz.com име='Ян' фамилия='Ковалски'

**Инсталиране и импортиране**на библиотеката със заявки. Можем да инсталираме библиотеката без никакви проблеми с помощта на pip:

pip заявки за инсталиране и можем да го импортираме също толкова лесно.

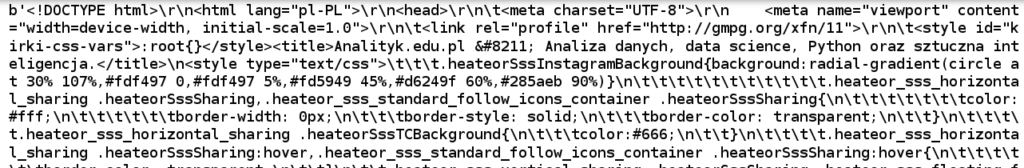
За тази цел използваме функцията GET. Status\_code, връща информация дали операцията е била успешна. Код 200 означава да. В противен случай имаме работа с грешка. Стойността 404 е добре позната стойност, която показва, че страницата не съществува.

След това можем да използваме следните функции, за да видим заглавката и съдържанието на страницата:

req.headers



req.text



**Пример за използване на функцията GET**

Като прост пример за използване на функцията get ще използваме dictionary.com. Тази услуга връща определението на думата, която сме въвели. Например, ако извършим GET операция на адрес https://www.dictionary.com/browse/computer, тогава в отговор ще получим html страница с определение на думата компютър.

Можем също да въведем този адрес в нашия браузър и да видим резултата. Ще го направим по-долу в Python:

Search = 'computer'

res = request. get ( 'https://www.dictionary.com/browse/' + search)

print ( res.url )

print ( res.text )

 В отговор ще получим:

Картина, която съдържа текст

Описанието е генерирано автоматично

В допълнение към определението на думата компютър има и много HTML маркиране и излишно съдържание. За целите на демонстрацията ще използваме още една библиотека, която ще ни улесни в извличането на това, което ни интересува. Ще обсъдим самата библиотека друг път, така че ще пропуснем подробностите тук:

From *bs4* importBeautifulSoup

soup = BeautifulSoup ( res.text, 'html.parser' )

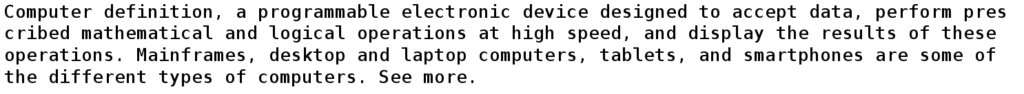
desc = soup. find\_all ( "meta" )

for label in soup. find\_all ( "meta" ) :

if label. get ( "name" ) == "description" :

print( receiving ( "content" ))

 И в отговор получаваме:



**Допълнителни параметри**

Съвсем скоро ще се сблъскаме с необходимостта от предаване на допълнителни параметри. Независимо дали е името на града, от началото на статията или API ключа.

За целите на упражненията ще използваме уебсайта https://httpbin.org/, който е създаден, за да упражним практически всички операции, които можем да извършваме с HTTP протокола. Описание на работата му може да се намери на уебсайта му.

 Допълнителни параметри могат да бъдат предадени под формата на речник към функцията GET. Нека разгледаме следния пример:

параметри = { 'masto' : 'Варшава' , 'страна' : 'Полша' }

r = request. get ( 'https://httpbin.org/get' , params=parameters )

print ( r.text )

 И в резултат услугата httpbin ще върне:

Картина, която съдържа текст, писмо

Описанието е генерирано автоматично

**Упълномощаване**

 В допълнение към предаването на допълнителни параметри, може да се наложи да се справим с необходимостта да разрешим връзката. Понякога това ще стане чрез предаване на API ключа в допълнителни параметри, а понякога, например, както в случая с gitHub, потребителско име и парола.

 Методът GET ни позволява да разрешим този проблем с помощта на параметъра auth=(login, password). Нека видим:

res = request. get ( 'https://api.github.com/user' , auth= ( 'login' , 'password' ))

print ( res.text )

 В отговор ще получим



 Въпреки това, когато предоставим истинското потребителско име и правилната парола, комуникацията ще бъде успешна.

 Друга функция, която си струва да знаете, е POST. Това води до изпращане на данни на посочения адрес. Работи подобно на функцията GET за допълнителни параметри, но вместо „params“ използваме променлива, наречена „data“.

body\_data = { "Възраст" : 30 , "Човек" : "Смит" }

r = request. post ( 'https://httpbin.org/post' , data=body\_data )

print ( r.text )

Картина, която съдържа текст

Описанието е генерирано автоматично

# Django REST Framework

Django REST Framework ( изходен код ), обикновено съкратено "DRF", е библиотека на Python за изграждане на интерфейси за програмиране на уеб приложения (API) .

Много рамки ви позволяват лесно да създавате API за блог приложения, но ние ще използваме само една – Django REST framework (DRF).

Той е удобен по много начини и предлага следните предимства:

Неговият API за уеб сърфиране е огромна полза за вашите разработчици.

Правилата за удостоверяване включват пакети за OAuth1 и OAuth2 .

Сериализацията поддържа както ORM , така и не-ORM източници на данни.

Може да се персонализира докрай. Просто използвайте обикновени изгледи, базирани на функции, ако не се нуждаете от по- мощните функции .

Има богата документация и голяма подкрепа от общността .

Използва се и се ползва с доверие от международно признати компании, включително Mozilla , Red Hat , Heroku и Eventbrite .

REST, което означава „REpresentational State Transfer“, е една от програмните архитектури, които определят как работят Django API. Терминът „RESTful“ се използва за уеб услуги, които се придържат към ограниченията на REST. С две думи, докато DRF улеснява работата с Django, RESTful API работят според REST.

# Django

Django е уеб рамка на високо ниво на Python, която насърчава бързото развитие и чистия, прагматичен дизайн. Създаден от опитни разработчици, той се грижи за голяма част от неприятностите на уеб разработката, така че можете да се съсредоточите върху писането на вашето приложение, без да е необходимо да преоткривате колелото. Той е безплатен и с отворен код.

Django е широко използвана рамка за уеб приложения на Python с философия „включени батерии“. Принципът зад batteries-included е, че общата функционалност за изграждане на уеб приложения трябва да идва с рамката, вместо като отделни библиотеки.

# Как сами да създавате и импортирате модули?

Модулите са файлове с разширение .py, които съдържат Python код. Можете да посочите всеки такъв файл като модул. Можете да ги намерите в стандартната библиотека на Python. Освен това някои модули са заредени с Python по подразбиране. Можете също така да инсталирате необходимите модули, като използвате мениджъра на пакети pip. Можете също да пишете свои собствени модули в Python.

## Създаване и импортиране на модули

Процесът на писане на модули не се различава от създаването на всеки друг Python файл. Вътре може да има дефиниции на класове, функции, променливи, които след това можете да използвате в други програми на Python.

## Как се пише модул?

Първо трябва да внедрите среда за разработка. В него създайте файл с разширение .py, например hello.py. След това можете да го импортирате в друг файл. Вътре трябва да напишете функция:

Можете също да стартирате файла с командата:

В същата папка трябва да създадете файл main\_program.py. Трябва да импортирате създадения модул в него и след това да извикате функцията. Файловете трябва да се намират в една и съща директория, тъй като модулът не е вграден. В противен случай Python няма да може да го намери. Наберете:

След импортирането на модула трябва да извикате функцията, като посочите името на модула в пунктирана нотация. Можете също да импортирате модул като този:

И след това го наречете като world (). Сега трябва да стартирате програмата през командния ред:

И следните термини ще се появят във вашия терминал:

Сега ще добавим променлива към модула hello.py:

Използвайте функцията print (), за да извикате променливата в нашия файл main\_program.py:

Сега, когато програмата стартира, на екрана ще се появи следният надпис:

Нека се опитаме сега да заменим класа в hello.py. Нуждаем се от потребителски клас с атрибути настроение и име. Също така се нуждаем от функция, която извежда тези атрибути:

Сега трябва да добавите класа в дъното на файла main\_program.py. Наберете:

Извикването на потребителския клас с hello.User () ще осъществи достъп до атрибутите и функциите в пространството от имена на файла main\_program.py. За да направите това, jesse.tell\_me\_about\_the\_user () се изписва на последния ред, без да вика здравей. Без препратка към този модел също ще бъде възможно да се извика всеки атрибут на клас. Когато стартирате приложението, ще видите:

Не забравяйте, че модулите могат да прилагат код, а не само да съхраняват дефиниции. Ще пренапишем файла hello.py, за да видим как работи. Трябва само да приложите функцията world ():

Следващата стъпка е да премахнете всички редове от main\_program.py с изключение на import:

Сега стартирайте програмата main\_program.py и ще видите на екрана:

Какво стана? Модулът hello е реализирал функцията world (). Той беше предаден на файла main\_program.py, който се изпълнява при.

## Достъп до модули от друга папка

Можете да използвате модули едновременно в няколко програми или проекти. В такава ситуация запазването на всичко в една папка е проблематично. Най-лесно е да съхранявате файлове в отделни папки на проекта. Ако искате да приложите модул от един проект към друг, можете да го направите по няколко начина:

1) Добавяне на път

Можете да извикате пътя на модула, като използвате програмните файлове, които го използват. Това не е нищо повече от временно решение, което отговаря на фазата на развитие. Този подход няма да ви позволи да направите един модул достъпен за цялата ви система.

За да добавите пътя на модула към друг файл, трябва да импортирате sys с други модули, които са ви необходими в основния програмен файл. Можете да намерите модула sys в стандартната библиотека на Python. Необходимо е да се осигурят индивидуални системни параметри и функции, чрез които ще бъде възможно да се зададе пътя на желания модул.

Например създадохте файл hello.py в папката/user/some /. В същото време все още имате файла main\_program.py в старата директория. За да импортирате модула hello.py в main\_program.py, трябва да прехвърлите sys, като добавите пътя/потребител/някои /. Използвайки го, Python ще може да намери посочения файл. Наберете:

Ако зададете правилно пътя към файла hello.py, ще можете да стартирате main\_program.py без грешки. Ще имаме същото заключение като преди.

2) Добавете модул към пътя на Python

Тази опция е по-удобна и може да се използва постоянно. Добавяте модул към пътя, от който Python получава пакети и модули. Вашият модул ще стане достъпен в цялата система. Първо трябва да разберете пътя. За целта стартирайте интерпретатора на Python в средата за разработка:

* Импортиране на модула sys:
* Сега трябва да покажете пътя на Python, като използвате функцията:

Тази команда ще отпечата поне един системен път. Въпреки това, в среда за разработка можете да получите няколко пътя наведнъж. В този случай ще трябва да определите правилния. Ще можете да добавяте модули към системната пътека на Python. За това:

Преместете файла hello.py в тази директория. След това импортирайте модула hello:

Ще видите, че програмата вече работи без грешки.

# Какво е HTTP?

[**HTTP**](https://bg.wikipedia.org/wiki/HTTP) (**H**yper**T**ext **T**ransfer **P**rotocol) е протокол за трансфер на [хипертекст](https://help.superhosting.bg/hypertex-world-wide-web.html) в интернет. Първата версия на протокола е от 1991 г., а през 2015 г. излиза новата и по-добра версия HTTP/2.

HTTP протоколът е измислен със създаването на [World Wide Web](https://help.superhosting.bg/world-wide-web.html) (световната информационна мрежа или уеб). Този протокол определя правилата, по които се извършва обмяната на информация между уеб сървър и уеб клиент (най-често уеб браузър).

Когато [Тим Бърнърс-Лий](https://bg.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BC_%D0%91%D1%8A%D1%80%D0%BD%D1%8A%D1%80%D1%81-%D0%9B%D0%B8%D0%B9) осъществява идеята си, за мрежа от взаимно свързана информация, наричана днес [уеб](https://help.superhosting.bg/world-wide-web.html), той създава и основните ѝ компоненти. Създаден е първият [уеб сървър](https://help.superhosting.bg/what-is-web-server.html), който предоставя първата уеб страница на първия уеб браузър. Комуникацията между уеб браузър и уеб сървър се извършва по създадения за тази цел протокол – HTTP.

В HTTP протокола [са описани](https://datatracker.ietf.org/doc/html/rfc7230) няколко [метода](https://datatracker.ietf.org/doc/html/rfc7231) за поискване (и изпращане) на информация от уеб браузъра към уеб сървъра. Тези HTTP методи се съдържат в заявките, които уеб браузърът изпраща към уеб сървъра.

Най-често срещаните HTTP методи са GET и POST. Съществуват още: HEAD, PUT, DELETE, TRACE, OPTIONS, CONNECT.

В HTTP протокола са описани и HTTP статус кодове, които се изпращат от уеб сървъра, в отговор на заявките от уеб браузъра.

## Уеб браузърът и уеб сървърът си говорят на един език – HTTP

Уеб браузърът се свързва с уеб сървъра чрез мрежова TCP връзка, по която изпраща HTTP заявки и приема HTTP отговори.

HTTP заявките от браузъра и HTTP отговорите на сървъра започват с тъй наречените HTTP хедъри. Това са редовете, съдържащи важните указания, по които протича HTTP обмена на информация.

В началото на HTTP заявката чрез HTTP хедърите браузърът описва какво търси и в какъв вид данни би могъл да го получи. Според това описание уеб сървърът ще знае какво и как точно да изпрати към уеб браузъра.

Например в следната HTTP заявка браузърът пита уеб сървъра за ресурса index.html, като го търси на хост help.superhosting.bg и заявява, че разговорът ще се проведе по протокол HTTP/1.1:

GET /index.html HTTP/1.1

Host: help.superhosting.bg

В следващите HTTP хедъри предоставя повече информация за себе си (User-Agent) и за възможностите и предпочитанията си. Подава информация на уеб сървъра, че може да приеме данните в определен формат (Accept) – text, html, xml, xhtml и че може да са компресирани (gzip, deflate).

User-Agent: Mozilla/5.0 Gecko/20100101 Firefox/51.0

Accept: text/html,application/xhtml+xml,application/xml

Accept-Encoding: gzip, deflate

На тази HTTP заявка уеб сървърът връща HTTP отговор, започващ с код 200 OK. Това означава, че търсеният ресурс е наличен, заявката е приета и информацията се изпраща към уеб браузъра.

HTTP/1.1 200 OK

В следващите HTTP хедъри уеб сървърът предоставя повече информация за себе си (Server), за установената връзка (Connection) и за самата информация (Content), която изпраща към браузъра.

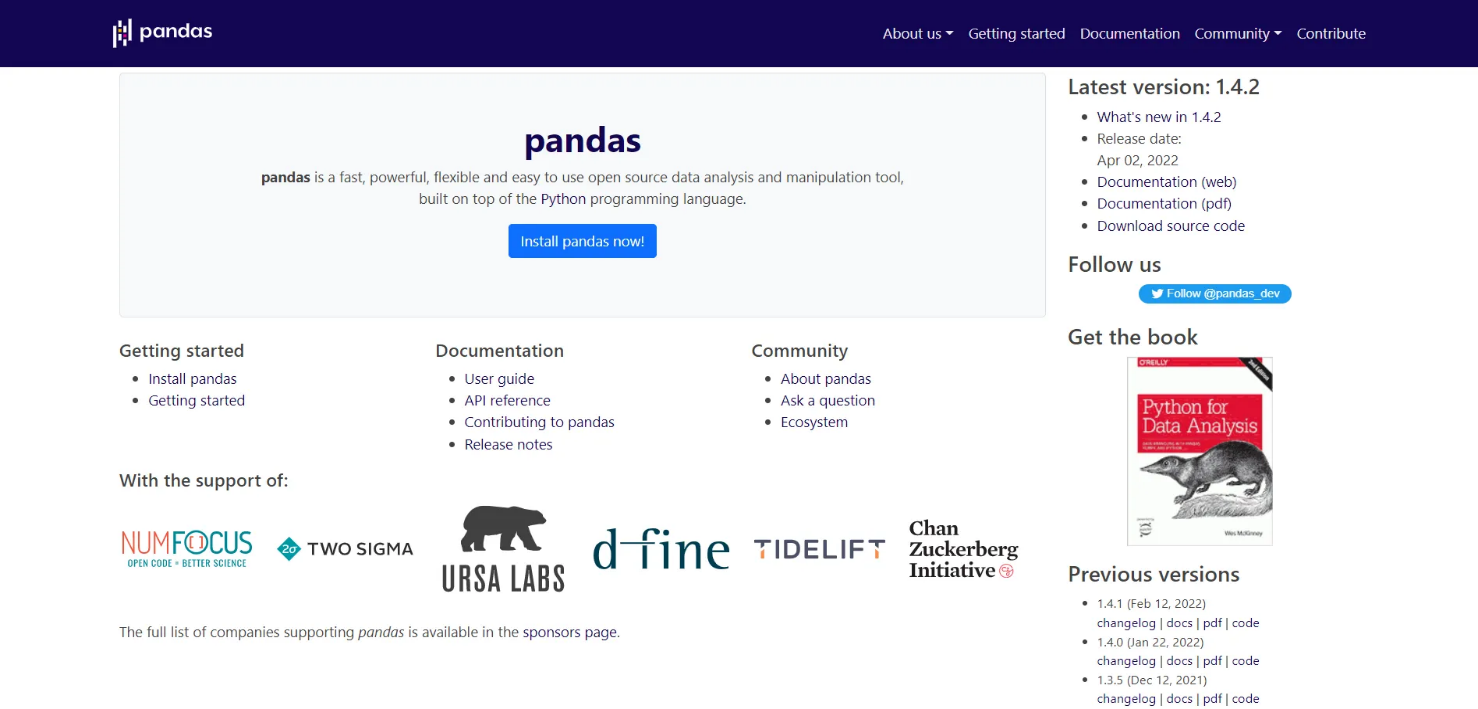
# Библиотеки на Python

Python никога не престава да учудва своите програмисти, когато става въпрос за преодоляване на предизвикателствата на науката за данните. Това е широко използван, обектно-ориентиран език за програмиране с отворен код и висока производителност с множество допълнителни функции.

Python е проектиран със забележителни библиотеки за наука за данни, които програмистите използват всеки ден за решаване на трудности.

Ето най-добрите библиотеки на Python, които да разгледате:

## Pandas

 Pandas е пакет, предназначен да подпомага разработчиците при работа с „маркирани“ и „релационни“ данни по естествен начин. Той е изграден върху две основни структури от данни: „Серии“ (едноизмерни, подобни на списък от обекти) и „Рамки с данни“ (двуизмерни, като таблица с множество колони).

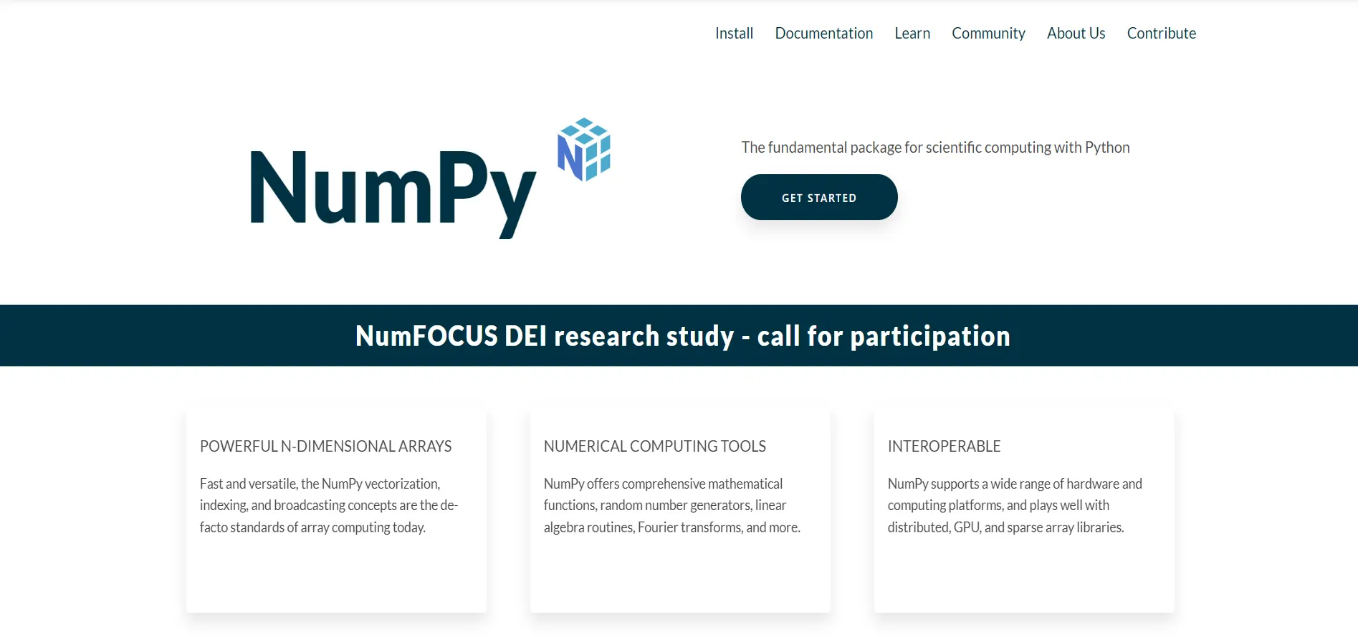
Pandas поддържа преобразуване на структури от данни в DataFrame обекти, справяне с липсващи данни, добавяне/изтриване на колони от DataFrame, импутиране на липсващи файлове и [визуализиране на данни](https://hashdork.com/bg/%D0%B8%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B8-%D0%B7%D0%B0-%D0%B2%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F-%D0%BD%D0%B0-%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8/) използване на хистограми или графики. Той също така предоставя редица инструменти за четене и запис на данни между структурите на данни в паметта и няколко файлови формата.

С две думи, той е идеален за бърза и проста обработка на данни, агрегиране на данни, четене и запис на данни и визуализация на данни. Когато създавате проект за наука за данни, винаги ще използвате библиотеката Pandas, за да обработвате и анализирате вашите данни.

## NumPy (Numerical Python)

NumPy (Numerical Python) е фантастичен инструмент за извършване на научни изчисления и основни и сложни операции с масиви.

Библиотеката предоставя редица полезни функции за работа с n-масиви и матрици в Python.



Улеснява обработката на масиви, които съдържат стойности от един и същ тип данни и извършването на аритметични операции върху масиви (включително векторизация). В действителност използването на типа масив NumPy за векторизиране на математически операции подобрява производителността и намалява времето за изпълнение.

Поддръжката на многомерни масиви за математически и логически операции е основната функция на библиотеката. Функциите на NumPy могат да се използват за индексиране, сортиране, преоформяне и предаване на визуални и звукови вълни като многоизмерен масив от реални числа.

## Python Matplotlib

В света на Python Matplotlib е една от най-широко използваните библиотеки. Използва се за генериране на статични, анимирани и интерактивни визуализации на данни. Matplotlib има много опции за диаграми и персонализиране.

Използвайки хистограми, програмистите могат да разпръскват, настройват и редактират графики. Библиотеката с отворен код предоставя обектно-ориентиран API за добавяне на диаграми в програми.

Когато обаче използват тази библиотека за генериране на сложни визуализации, разработчиците трябва да напишат повече код от нормалното.



Заслужава да се отбележи, че популярните библиотеки за диаграми съжителстват с Matplotlib без проблеми.

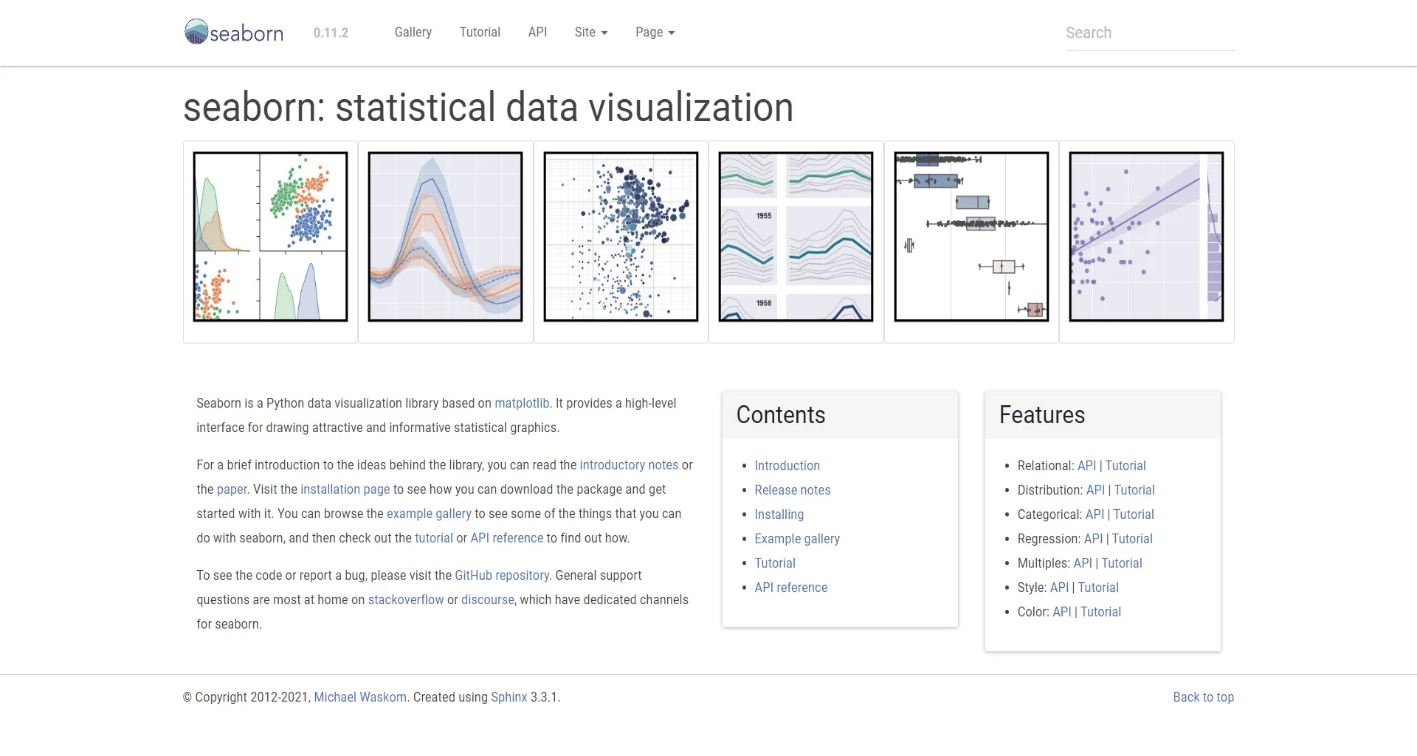
Освен всичко друго, той се използва в скриптове на Python, обвивки на Python и IPython, преносими компютри на Jupyter и [уеб приложение](https://hashdork.com/bg/%D0%BD%D0%B0%D0%B9-%D0%B4%D0%BE%D0%B1%D1%80%D0%B8%D1%82%D0%B5-%D0%B7%D0%B0%D1%89%D0%B8%D1%82%D0%BD%D0%B8-%D1%81%D1%82%D0%B5%D0%BD%D0%B8-%D0%B7%D0%B0-%D1%83%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F/) сървъри.

Графики, лентови диаграми, кръгови диаграми, хистограми, точкови диаграми, диаграми на грешките, спектри на мощността, стълбови диаграми и всякакви други видове диаграми за визуализация могат да бъдат създадени с него.

## Seaborn

Библиотеката Seaborn е изградена на Matplotlib. Seaborn може да се използва за създаване на по-привлекателни и информативни статистически графики от Matplotlib.

Seaborn включва интегриран API, ориентиран към набор от данни за изследване на взаимодействията между много променливи, в допълнение към пълната поддръжка за визуализация на данни.

Seaborn предлага зашеметяващ брой опции за визуализация на данни, включително визуализация на времеви серии, съвместни диаграми, диаграми на цигулка и много други.

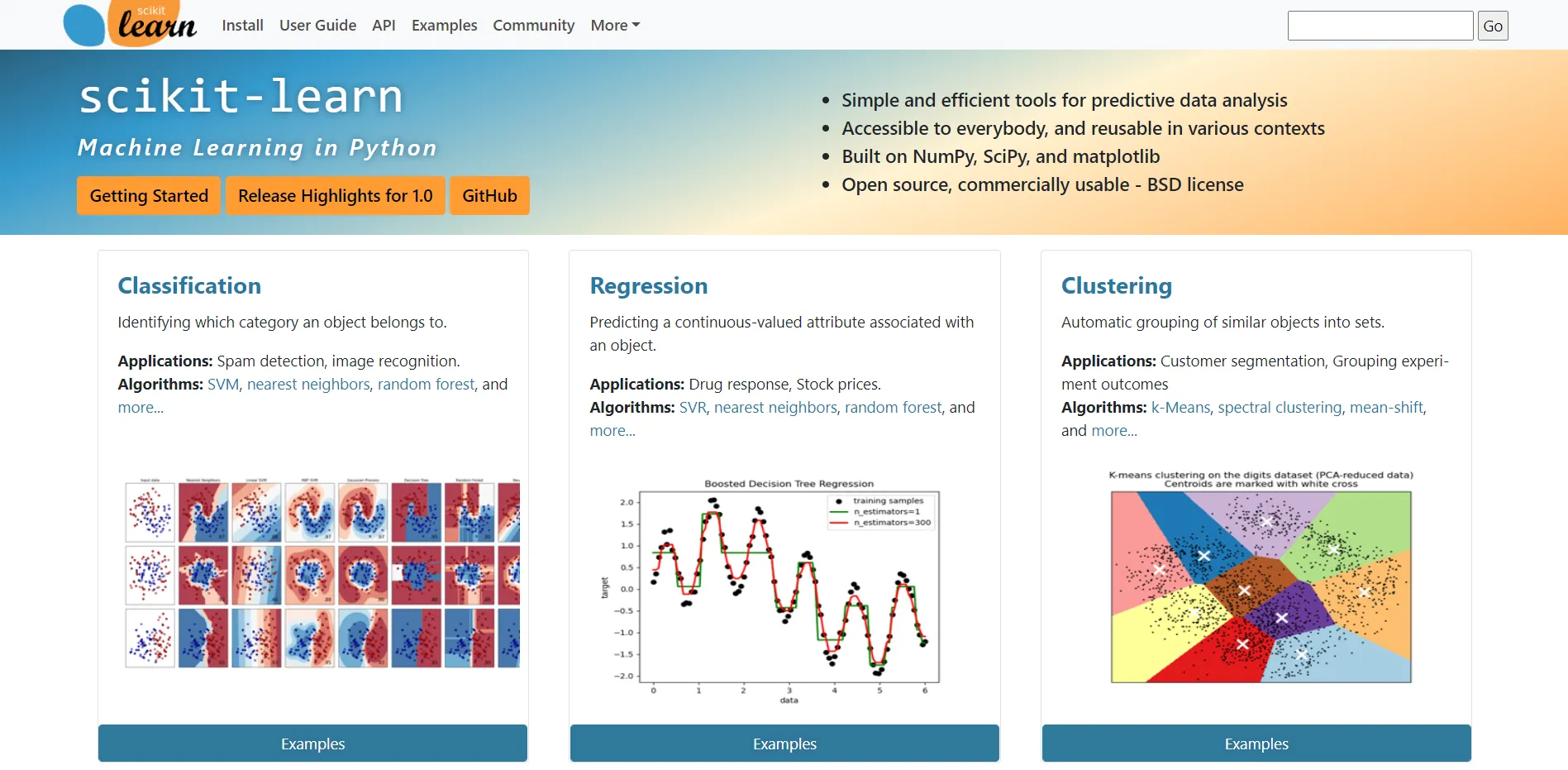
Той използва семантично картографиране и статистическо агрегиране, за да предостави информативни визуализации с дълбоки прозрения. Той включва редица рутинни процедури за диаграми, ориентирани към набор от данни, които работят с рамки от данни и масиви, които включват цели набори от данни.

Неговите визуализации на данни могат да включват стълбови диаграми, кръгови диаграми, хистограми, точкови диаграми, диаграми на грешки и други графики. Тази библиотека за визуализация на данни на Python също включва инструменти за избор на цветови палитри, които помагат при разкриването на тенденции в набор от данни.

## Scikit-learn

Scikit-learn е най-голямата библиотека на Python за моделиране на данни и оценка на модела. Това е една от най-полезните библиотеки на Python. Той има множество възможности, предназначени единствено за целите на моделирането.

Той включва всички контролирани и неконтролирани алгоритми за машинно обучение, както и напълно дефинирани функции за обучение на ансамбъл и усилване на машинно обучение.



Използва се от специалисти по данни за извършване на рутина [машинно обучение](https://hashdork.com/bg/%D0%BD%D0%B0%D0%B1%D0%BE%D1%80%D0%B8-%D0%BE%D1%82-%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8-%D0%B7%D0%B0-%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE-%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5/) и дейности по извличане на данни като групиране, регресия, избор на модел, намаляване на размерността и класификация. Освен това идва с изчерпателна документация и се представя чудесно.

Scikit-learn може да се използва за създаване на различни контролирани и неконтролирани модели на машинно обучение като класификация, регресия, опорни векторни машини, произволни гори, най-близки съседи, наивни Bayes, дървета на решенията, групиране и т.н.

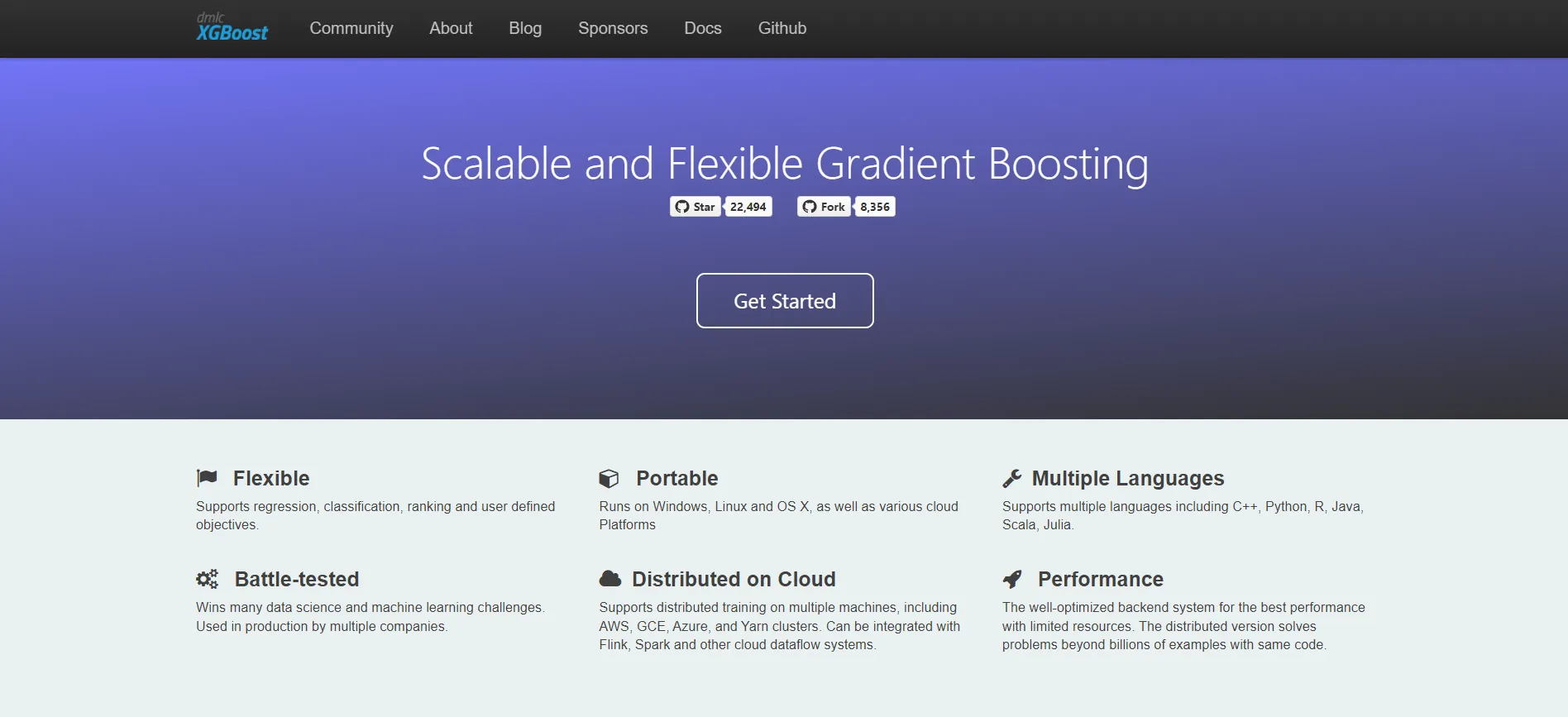
Библиотеката за машинно обучение на Python включва разнообразие от прости, но ефективни инструменти за извършване на анализ на данни и задачи за копаене.

За допълнително четене, ето нашето ръководство за[Scikit-учете.](https://hashdork.com/bg/scikit-%D0%BD%D0%B0%D1%83%D1%87%D0%B5%D1%82%D0%B5/)

## XGBoost

XGBoost е разпределен набор от инструменти за усилване на градиент, предназначен за скорост, гъвкавост и преносимост. За разработване на ML алгоритми, той използва рамката Gradient Boosting. XGBoost е бърза и точна техника за усилване на паралелно дърво, която може да реши широк спектър от проблеми с науката за данни.

Използвайки рамката Gradient Boosting, тази библиотека може да се използва за създаване на алгоритми за машинно обучение.



Той включва паралелно увеличаване на дървото, което помага на екипите при решаването на различни проблеми с науката за данни. Друго предимство е, че разработчиците могат да използват един и същ код за Hadoop, SGE и MPI.

Освен това е надежден както в разпределени, така и в ситуации с ограничена памет.

## TensorFlow

TensorFlow е безплатна AI платформа с отворен код от край до край с широк набор от инструменти, библиотеки и ресурси. TensorFlow трябва да е познат на всеки, който работи [проекти за машинно обучение](https://hashdork.com/bg/%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8-%D0%B7%D0%B0-%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE-%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5-%D0%B7%D0%B0-%D0%BD%D0%B0%D1%87%D0%B8%D0%BD%D0%B0%D0%B5%D1%89%D0%B8/) в Python.

Това е символичен математически инструментариум с отворен код за числени изчисления, използващи графики на потока от данни, разработени от Google. Възлите на графиката отразяват математическите процеси в типична графика на потока от данни TensorFlow.

Ръбовете на графиката, от друга страна, са многомерните масиви от данни, известни също като тензори, които протичат между мрежовите възли.

Той позволява на програмистите да разпределят обработката между един или повече процесори или графични процесори на десктоп, мобилно устройство или сървър, без да променят кода.

Картина, която съдържа диаграма

Описанието е генерирано автоматично

TensorFlow е разработен на C и C++. С TensorFlow можете просто да проектирате и [тренирайте машинно обучение](https://hashdork.com/bg/%D0%BE%D0%B1%D1%8F%D1%81%D0%BD%D0%B5%D0%BD%D0%BE-%D0%B7%D0%B0%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B5-%D0%B7%D0%B0-%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5-%D0%B7%D0%B0-%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE-%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5/) модели, използващи API на високо ниво като Keras.

Освен това има много степени на абстракция, което ви позволява да изберете най-доброто решение за вашия модел. TensorFlow също ви позволява да разположите модели на машинно обучение в облака, браузър или вашето собствено устройство.

Това е най-ефективният инструмент за задачи като разпознаване на обекти, разпознаване на реч и много други. Подпомага развитието на изкуствени [невронни мрежи](https://hashdork.com/bg/%D0%BD%D0%B5%D0%B2%D1%80%D0%BE%D0%BD%D0%BD%D0%B0-%D0%BC%D1%80%D0%B5%D0%B6%D0%B0/) които трябва да работят с множество източници на данни.

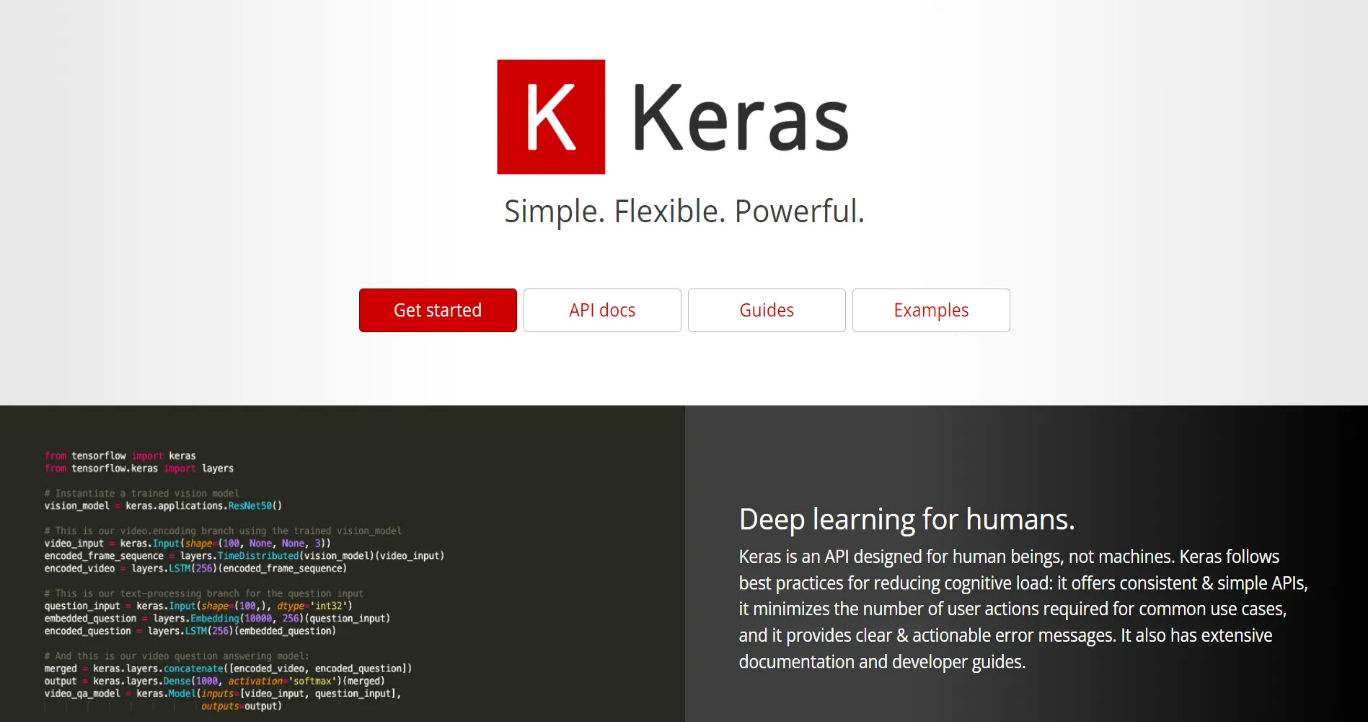
[Ето нашето кратко ръководство за TensorFlow за допълнително четене.](https://hashdork.com/bg/tensorflow-%D1%80%D0%B0%D0%BC%D0%BA%D0%B0-%D0%B7%D0%B0-%D0%B4%D1%8A%D0%BB%D0%B1%D0%BE%D0%BA%D0%BE-%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5/)

## [Keras](https://keras.io/" \t "_blank)

Keras е безплатен и с отворен код [Невронна мрежа, базирана на Python](https://hashdork.com/bg/%D0%B8%D0%B7%D0%B3%D1%80%D0%B0%D0%B6%D0%B4%D0%B0%D0%BD%D0%B5-%D0%BD%D0%B0-%D0%BD%D0%B5%D0%B2%D1%80%D0%BE%D0%BD%D0%BD%D0%B8-%D0%BC%D1%80%D0%B5%D0%B6%D0%B8-%D0%B2-python/) набор от инструменти за дейности с изкуствен интелект, задълбочено обучение и наука за данни. Невронните мрежи също се използват в Data Science за интерпретиране на данни от наблюдения (снимки или аудио).

Това е колекция от инструменти за създаване на модели, графично изобразяване на данни и оценка на данни. Той също така включва предварително маркирани набори от данни, които могат бързо да бъдат импортирани и заредени.

Той е лесен за използване, универсален и идеален за проучвателни изследвания. Освен това ви позволява да създавате напълно свързани, конволюционни, обединяващи, повтарящи се, вграждащи и други форми на невронни мрежи.



Тези модели могат да бъдат обединени, за да се изгради пълноценна невронна мрежа за огромни набори от данни и проблеми. Това е фантастична библиотека за моделиране и създаване на невронни мрежи.

Той е лесен за използване и дава на разработчиците голяма гъвкавост. Keras е бавен в сравнение с други пакети за машинно обучение на Python.

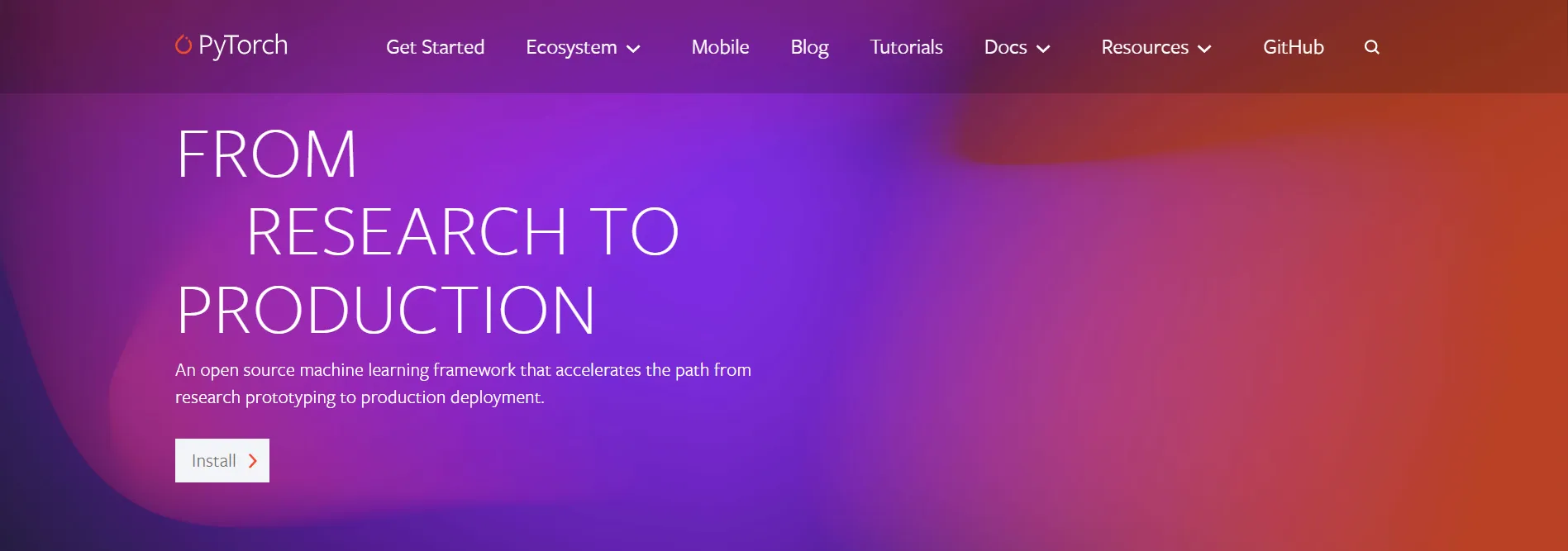
Това е така, защото първо генерира изчислителна графика, използвайки бекенд инфраструктурата и след това я използва за извършване на операции. Keras е невероятно изразителен и адаптивен, когато става въпрос за извършване на нови изследвания.

## PyTorch

PyTorch е популярен пакет на Python за [дълбоко учене](https://hashdork.com/bg/%D1%80%D0%B0%D0%BC%D0%BA%D0%B8-%D0%B7%D0%B0-%D0%B4%D1%8A%D0%BB%D0%B1%D0%BE%D0%BA%D0%BE-%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5/) и машинно обучение. Това е базиран на Python научен компютърен софтуер с отворен код за внедряване на задълбочено обучение и невронни мрежи върху огромни набори от данни.

Facebook широко използва този инструментариум за създаване на невронни мрежи, които подпомагат дейности като лицево разпознаване и автоматично маркиране.

PyTorch е платформа за учени по данни, които желаят бързо да завършат задачи за дълбоко обучение. Инструментът позволява тензорни изчисления да се извършват с GPU ускорение.



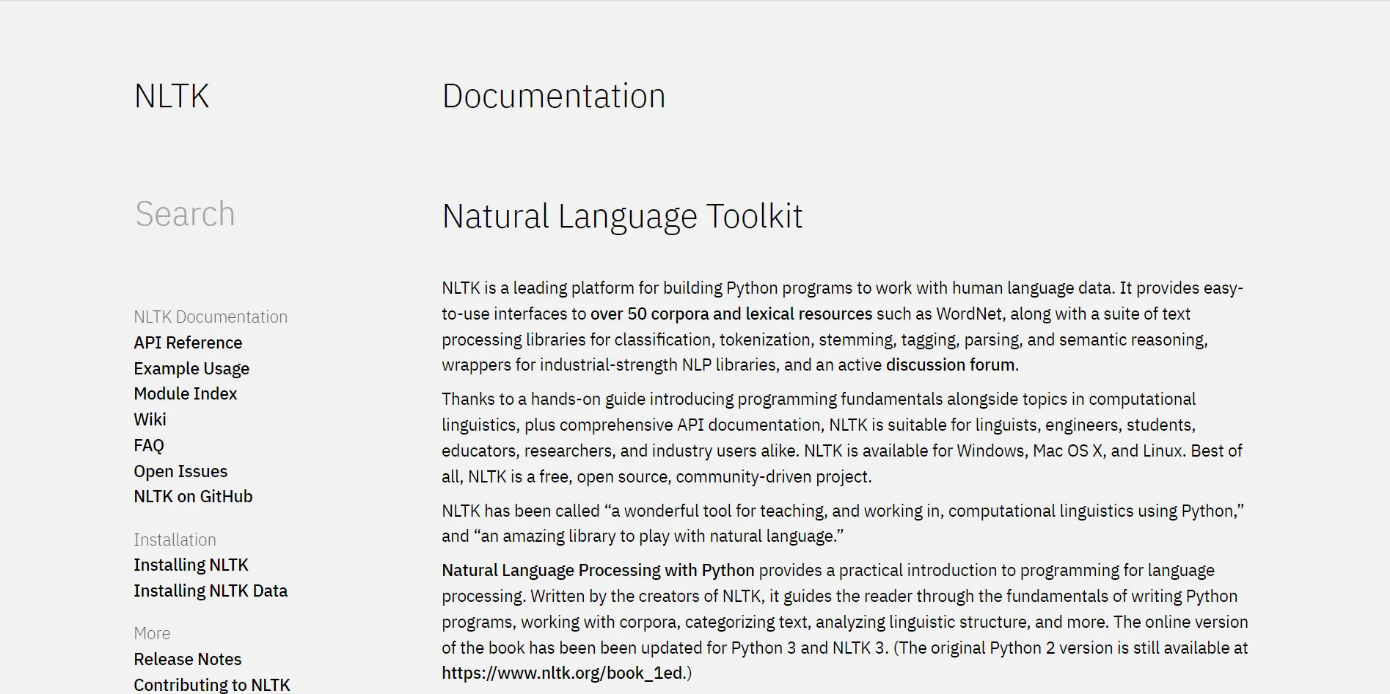
Използва се и за други неща, включително изграждане на динамични изчислителни мрежи и автоматично изчисляване на градиенти.

За щастие, PyTorch е фантастичен пакет, който позволява на разработчиците лесно да преминат от теория и изследвания към обучение и развитие, когато става въпрос за машинно обучение и дълбоко обучение, за да осигурят максимална гъвкавост и скорост.

## NLTK (Natural Language Toolkit)

NLTK (Natural Language Toolkit) е популярен пакет на Python за специалисти по данни. Маркиране на текст, токенизация, семантично разсъждение и други задачи, свързани с обработката на естествен език, могат да бъдат изпълнени с NLTK.

NLTK може да се използва и за завършване на по-сложен AI ([Изкуствен интелект](https://hashdork.com/bg/%D0%BA%D0%B0%D0%BA%D0%B2%D0%BE-%D0%B5-%D0%B8%D0%B7%D0%BA%D1%83%D1%81%D1%82%D0%B2%D0%B5%D0%BD-%D0%B8%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82/)) работни места. Първоначално NLTK е създаден, за да поддържа различни парадигми за преподаване на AI и машинно обучение, като лингвистичния модел и когнитивната теория.



Понастоящем управлява алгоритъма за AI и разработването на модел за обучение в реалния свят. Той е широко възприет за използване като учебен инструмент и като инструмент за индивидуално обучение, в допълнение към използването му като платформа за създаване на прототипи и разработване на изследователски системи.

Поддържат се класификация, синтактичен анализ, семантично разсъждение, произтичане, маркиране и токенизиране.

# Обяснение на кода

import os

import glob

# get all files inside a specific folder

dir\_path = r'C:\ggb\GEORGI\DZI\_22\_23'

for path in os.scandir(dir\_path):

if path.is\_file():

print(path.name)

# search all files inside a specific folder

# \*.\* means file name with any extension

dir\_path = r'C:\ggb\GEORGI\DZI\_22\_23\\*.\*'

res = glob.glob(dir\_path, recursive=True)

print(res)

for path in res:

print(path)

* Кодът започва с импортиране на os модула.
* След това кодът получава пътя за папката, която съдържа GEORGI\DZI\_22\_23 и използва scandir(), за да намери всички файлове в нея.
* Това се използва за получаване на списък с всички файлове в конкретна папка.
* След това кодът преминава през всеки файл и отпечатва името му, ако е файл, в противен случай отпечатва „Няма“.
* Следващият ред получава всички файлове в папката C:\ggb\GEORGI\DZI\_22\_23 с произволно разширение.
* Той прави това с помощта на glob(), който приема два аргумента: dir\_path и recursive=True.
* След това glob() се използва за търсене във всички тези имена на файлове и отпечатването им, както и пътищата им на екрана.
* Кодът ще отпечата всички файлове в папка C:\ggb\GEORGI\DZI\_22\_23.

# Обяснение на клиентския код

import ftplib

# set up connection parameters

ftp\_server = '192.168.99.103'

ftp\_port = 2121

ftp\_username = 'USER3'

ftp\_password = '2121'

# connect to FTP server

ftp = ftplib.FTP()

ftp.connect(ftp\_server, ftp\_port)

ftp.login(ftp\_username, ftp\_password)

# list files on server

print('Files on server:')

files = ftp.dir()

print(files)

with open('test.txt', 'w') as f:

f.write('This is a test file.')

# upload file to server

filename = 'test.txt'

with open(filename, 'rb') as f:

ftp.storbinary(f'STOR {filename}', f)

# download file from server

filename = 'test.txt'

with open(filename, 'wb') as f:

ftp.retrbinary(f'RETR {filename}', f.write)

# close connection

ftp.quit()

* Кодът започва с импортиране на библиотеката ftplib.
* След това кодът задава параметри за връзка за FTP сървъра и потребителско име/парола.
* След като се свърже с FTP сървъра, той влиза с тези идентификационни данни.
* Той изброява файловете на сървъра с оператор за печат, който е последван от файлов обект, който съдържа информация за всеки файл в списъка.
* След това отваря текстов файл и записва малко съдържание в него, преди да качи този текстов файл на FTP сървъра с извикване на функция open().
* Следващият ред от код използва ftp обект от ftplib, за да качи test.txt от файловата система на вашия компютър във файловата система на вашата локална машина чрез FTP (протокол за прехвърляне на файлове).
* Това се прави с метода storbinary() на ftp обект, който приема два аргумента: име на файл и f.
* В този случай предавате „тест“ като име на файл и пишете („Това е тестов файл.“)
* като f .
* Резултатът от тази команда ще бъде записан обратно на вашата локална машина, когато по-късно стартирате отново скрипта на python, за да го изтеглите обратно от файловата система на вашата локална машина чрез FTP (протокол за прехвърляне на файлове) с извикване на функция open() като така : ftp = ftplib.FTP() ftp.connect
* Кодът ще качи файла "test.txt" на сървъра и ще го изтегли обратно.

# Списък на използваната литература

1. https://hashdork.com/bg/python-libraries-for-data-scientists/
2. https://blog.superhosting.bg/why-and-for-what-python.html
3. https://help.superhosting.bg/http-hypertext-transfer-protocol.html
4. https://docs-python.ru/packages/modul-requests-python/
5. https://analityk.edu.pl/python-request-biblioteka-do-wykonywania-zapytan-http/
6. https://docs.python.org/3/library/http.server.html