Съдържание

[1 Увод: 3](#_Toc133574398)

[2 Python 3](#_Toc133574399)

[*2.1* *Къде е приложим Python* 4](#_Toc133574400)

[2.2 *Защо Python е толкова широко разпространен* 4](#_Toc133574401)

[*2.3* *За какво се използва Python* 5](#_Toc133574402)

[2.3.1 Какво можете да правите с Python? 5](#_Toc133574403)

[2.3.2 Софтуерно тестване и прототипиране: 5](#_Toc133574404)

[2.3.3 Ежедневни задачи: 6](#_Toc133574405)

[2.3.4 Автоматизация или скриптове: 6](#_Toc133574406)

[2.3.5 Уеб разработка: 6](#_Toc133574407)

[2.4 Какво е Moviepy 7](#_Toc133574408)

[2.5 Анализ на данни и машинно обучение: 7](#_Toc133574409)

[*2.6* *Как Python обработва файлове* 7](#_Toc133574410)

[2.6.1 Потребителски интерфейси на Python 8](#_Toc133574411)

[2.6.2 Налични платформи: 8](#_Toc133574412)

[*2.7* *Мислите ли как python ще обработва файлове?* 11](#_Toc133574413)

[*3* *Що е алгоритъм?* 11](#_Toc133574414)

[3.1 Представяне: 14](#_Toc133574415)

[3.1.1 Описание от високо ниво: 14](#_Toc133574416)

[3.1.2 Описание на реализацията: 15](#_Toc133574417)

[3.1.3 Формално описание: 15](#_Toc133574418)

[4 PyCharm: 15](#_Toc133574419)

[*4.1* *Характеристики:* 16](#_Toc133574420)

[4.1.1 Pycharm: среда за разработка на Python: 16](#_Toc133574421)

[4.1.2 Python рефакторинг: 17](#_Toc133574422)

[*4.2* *Как да инсталираме PyCharm на Linux?* 17](#_Toc133574423)

[4.2.1 Първоначална настройка на Pycharm: 17](#_Toc133574424)

[5 Видео файлове: 18](#_Toc133574425)

[5.1 Анатомия на видео файла: 19](#_Toc133574426)

[5.1.1 Разделителна способност: 19](#_Toc133574427)

[5.2 *Какво е Gizeh* 20](#_Toc133574428)

[5.2.1 Какво ни трябва? 20](#_Toc133574429)

[5.2.2 Честота на кадрите: 21](#_Toc133574430)

[5.2.3 Скорост: 22](#_Toc133574431)

[5.2.4 Кодеци: 23](#_Toc133574432)

[5.3 Необработени видео формати: 24](#_Toc133574433)

[5.4 Компресирани видео формати: 25](#_Toc133574434)

[5.5 Добавяне на файл с изображение: 28](#_Toc133574435)

[*6* *Използвани библиотеки за създаване на програмата:* 29](#_Toc133574436)

[6.1 CV2 29](#_Toc133574437)

[6.2 Pyautogui 30](#_Toc133574438)

[6.3 Сейфове: 32](#_Toc133574439)

[7 Списък на използваната литература: 34](#_Toc133574440)

# Увод:

Видео файлът съхранява визуална, звукова и друга информация. Категорията Видео файлове съдържа голямо разнообразие от видео формати, които използват различни кодеци, за да кодират и компресират различните типове видео. Моята задача е да направя приложение за запис във видео файл на работен сеанс с дадено приложение. Избрах езикът за програмиране да бъде Python, който е от високо ниво и с много широко приложение. Той се използва за разработване на уебсайтове, софтуер и най-вече има висока степен на четимост, което го прави достъпен за много хора. Библиотеките, които избрах са OpenCv и Pyautogui. OpenCV е библиотека от програмни функции главно за компютърно зрение в реално време. Библиотеката е междуплатформена и е лицензирана като безплатен софтуер с отворен код под Apache, също включва GPU ускорение за операции в реално време. Другата библиотека е PyAutoGUI, тя позволява на вашите Python скриптове да контролират мишката и клавиатурата, за да автоматизират взаимодействията с други приложения. Тя работи на Windows и Linux. Може би сте чували за ImageMagick за издаване на изображения или видео. MoviePy е Python модул за редактиране на видео и предоставя функции за рязане, вмъкване на заглавия и обработка на видео. Броя на неподвижни изображения, които се появяват едно след друго в рамките на една секунда,за да създадат движение се нарича честота на кадрите. Колкото е по-висока скоростта на кадрите, толкова по-плавно и естествено изглежда движението. Минималната скорост, с която човек възприема изображението гладко е 16 кадъра в секунда, а аз съм го направила да бъде 30 кадъра в секунда, а времето на продължителността на записа ми е 10 секунди. Смятам, че приложението ми може да бъде използвано в ежедневието, за преподаване, за запис на работа, която предстои да бъде отложена и много други.

# Python

Python е език за програмиране от високо ниво, интерпретативен, интерактивен, обектно ориентиран, създаден от Гуидо ван Росум в началото на 90-те години. Кръстен е на телевизионното шоу на BBC Monty Python’s Flying Circus. Често бива сравняван с Tcl, Perl, Scheme, Java и Ruby. Python има сходства с езици като Perl, C, C++ и Java. Той е език за програмиране с общо предназначение. Други негови отличителни черти са, че е обектно-ориентиран, интерпретируем и интерактивен. „Интерпретируем“ означава, че за да изпробвате даден код дали работи, не е нужно да го компилирате преди това. „Обектно-ориентиран“ – че Python поддържа технологията на програмиране, при която кода се обособява в групи, наречени „обекти“, които взаимодействат помежду си.

## Къде е приложим Python

В наши дни този език се прилага навсякъде: от 3D рендериране във филмовата или игралната индустрия, управление и автоматизация на сървъри, дори и програми за други операционни системи като Linux OS. Други приложения като Drop Box, Bit Torrent, Blender също са написани на Python. За да направят софтуера си по-бърз, редица компании използват Python. Facebook и NASA са две от най-популярните имена, които използват Python широко в своята дейност. Python e често използван за разработка на уебсайт – софтуерната рамка на Python за разработка на уеб сайтове – “Django”, е една от най използваните в наши дни. Или може би желаете да разработвате игри? Тогава кросплатформения набор от Python модули (Pygame) е това, от което имате нужда. Ако искате да разработвате заглавия за мобилни платформи, то тогава имате на разположение библиотеката с отворен код Kivy. Python намира приложение и в машинното обучение (machine learning), съответно и в разработките на изкуствен интелект.

## Защо Python е толкова широко разпространен

Най-вече защото кодът на Python има висока степен на четимост (т.е. почти идентичен с нормалния английски език), което го прави достъпен за много хора. Привличането на голям брой хора, които да го използват от своя страна увеличава способностите на езика, увеличава ресурсите, които са налични, увеличава проблемите, които могат да се решават с този език. Но най-вече Python никога не е зависел от платформата, за която се разработват приложенията. Дори да се върнем назад във времето и тръгнем към наши дни, можем да видим за колко различни платформи съществуват приложения, писани на Python: Linux, Window, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, Z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE.

## За какво се използва Python

Python обикновено се използва за разработване на уебсайтове и софтуер, автоматизация на задачи, анализ на данни и визуализация на данни. Тъй като е относително лесен за научаване, Python е възприет от много непрограмисти, като счетоводители и учени, за различни ежедневни задачи, като организиране на финанси. „Писането на програми е много креативна и възнаграждаваща дейност“, казва инструкторът от Мичиганския университет и Coursera Чарлз Р. Северънс в книгата си „ Python for Everybody“. „Можете да пишете програми по много причини, вариращи от изкарване на прехраната ви до решаване на труден проблем с анализ на данни до забавление до помагане на някой друг да реши проблем.“

### ****Какво можете да правите с Python?****

Някои неща включват:

• Анализ на данни и машинно обучение

• Уеб разработка

• Автоматизация или скриптове

• Софтуерно тестване и прототипиране

• Ежедневни задачи

### ****Софтуерно тестване и прототипиране:****

При разработването на софтуер Python може да помогне при задачи като контрол на изграждането, проследяване на грешки и тестване. С Python разработчиците на софтуер могат да автоматизират тестването за нови продукти или функции. Някои инструменти на Python, използвани за тестване на софтуер, включват Green и Requestium.

### ****Ежедневни задачи:****

Python не е само за програмисти и специалисти по данни. Изучаването на Python може да отвори нови възможности за тези в професии с по-малко данни, като журналисти, собственици на малък бизнес или специалисти по маркетинг в социални медии. Python може също да позволи на непрограмисти да опростят определени задачи в живота си. Ето само няколко от задачите, които можете да автоматизирате с Python:

• Следете цените на фондовия пазар или крипто

• Изпратете си текстово напомняне да носите чадър винаги, когато вали

• Актуализирайте списъка си за пазаруване на хранителни стоки

• Преименуване на големи пакети от файлове

• Преобразуване на текстови файлове в електронни таблици

• На случаен принцип възлагайте задължения на членове на семейството

• Попълвайте онлайн формуляри автоматично

### ****Автоматизация или скриптове:****

Ако установите, че изпълнявате задача многократно, можете да работите по-ефективно, като я автоматизирате с Python. Писането на код, използван за изграждане на тези автоматизирани процеси, се нарича скриптове. В света на кодирането автоматизацията може да се използва за проверка за грешки в множество файлове, конвертиране на файлове, изпълнение на проста математика и премахване на дубликати в данни. Python може да се използва дори от сравнително начинаещи за автоматизиране на прости задачи на компютъра - като преименуване на файлове, намиране и изтегляне на онлайн съдържание или изпращане на имейли или текстове на желани интервали.

### ****Уеб разработка:****

Python често се използва за разработване на задната част на уебсайт или приложение - частите, които потребителят не вижда. Ролята на Python в уеб разработката може да включва изпращане на данни до и от сървъри, обработка на данни и комуникация с бази данни, URL маршрутизиране и гарантиране на сигурност. Python предлага няколко рамки за уеб разработка. Често използваните включват Django и Flask. Някои работни места за уеб разработка, които използват Python, включват back end инженери, full stack инженери, Python разработчици, софтуерни инженери и DevOps инженери.

## Какво е Moviepy

Може би сте чували за FFMPEG или ImageMagick за издаване на изображения и видео по програмен начин. MoviePy е Python модул за редактиране на видео (Python обвивка за FFMPEG). Предоставя функции за рязане, конкатенации, вмъкване на заглавия, композиране на видео, обработка на видео и създаване на персонализирани ефекти. Може да чете и записва общи видео и аудио формати и да се изпълнява на всяка платформа с Python.

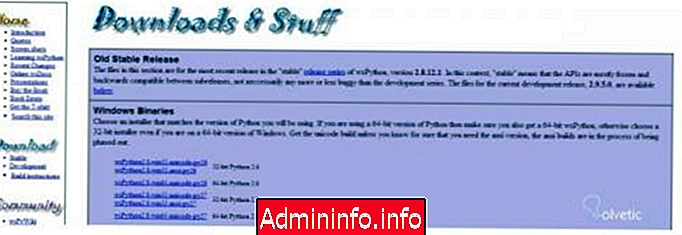
## ****Анализ на данни и машинно обучение:****

Python се превърна в основен елемент в науката за данни, позволявайки на анализатори на данни и други професионалисти да използват езика за извършване на сложни статистически изчисления, създаване на визуализации на данни, изграждане на алгоритми за машинно обучение, манипулиране и анализ на данни и изпълнение на други задачи, свързани с данни. Python може да създава широк набор от различни визуализации на данни, като линейни и лентови графики, кръгови диаграми, хистограми и 3D графики. Python също има редица библиотеки, които позволяват на програмистите да пишат програми за анализ на данни и машинно обучение по-бързо и ефективно, като TensorFlow и Keras.

## Как Python обработва файлове

Ако работите в голямо софтуерно приложение, където те обработват голям брой данни, тогава не можем да очакваме тези данни да се съхраняват в променлива, тъй като променливите са нестабилни по своята същност. Следователно кога ще се справяте с подобни ситуации, ролята на файловете ще се появи в картината. Тъй като файловете са енергонезависими по своята същност, данните ще се съхраняват постоянно във вторично устройство.

### Потребителски интерфейси на Python



Програма или приложение не винаги трябва да бъде нещо, което се изпълнява самостоятелно, всъщност много пъти изпълняваме програми, които трябва да се използват от хората, тези хора, които наричаме потребители; нашата програма може да е най-добрата и най-напредналата в света по своята логика и поведение, но ако е ориентирана да бъде за използване на потребител, начинът, по който може да взаимодейства със споменатата програма, е почти толкова или може би по-важен от самата логика. Поради това Python предлага възможност за използване на инструменти за изграждане на потребителски интерфейси, което ни позволява да дадем графичен смисъл на нашата програма, да организираме елементите, така че трета страна да може да я използва.

### Налични платформи:

Има много платформи за разработване на графични потребителски интерфейси в Python, тази, която избираме, ще бъде добре, стига да знаем защо го правим, нито една не е по-добра от друга, те са просто различни начини за постигане на цел.

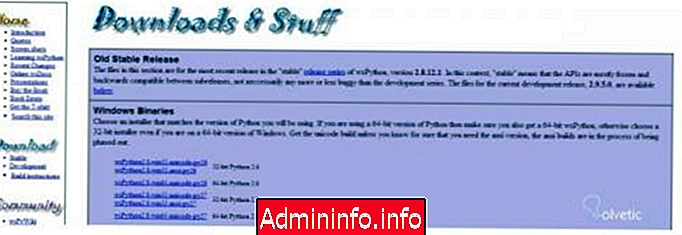
Нека видим някои от възможностите, които са на разположение, като поясним, че те не са всички, които съществуват, може да има и със сигурност ще има хиляди опции, но те могат да послужат като основа за започване от някакъв момент.

Тези инструменти се наричат инструменти, нека видим списъка:

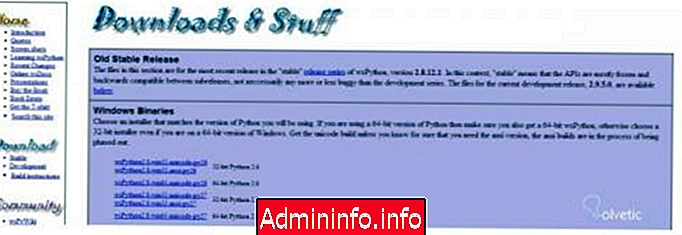
* Tkinter: Използвайте платформата TK. Има незабавна наличност. Тя е полустандартна.
* wxPython: Въз основа на wx Windows . Популярността му нараства много бързо.
* PythonWin: Достъпно само за Windows. Той използва родните възможности на средата на графичния интерфейс на Windows.
* Java Swing: Достъпно само за Jython . Той използва възможностите на родния графичен интерфейс на Java.
* PyGTK: Използвайте платформата GTK. Много популярен в Linux среди.
* PyQt: Използвайте Qt платформата. Той е особено популярен в Linux платформи.
* В случая на този урок ще използваме wxPython като демонстрация, за това ще го изтеглим от неговата уеб страница, достъпна на URL адрес //wxpython.org/download.php.

Важно е да изтеглите версията, която е направена за версията Python, която работим, за да не хвърля грешки.

На следващото изображение можем да видим версиите и как изглежда страницата за изтегляне на този инструментариум:



Веднъж изтеглени и конфигурирани, можем да преминем към следващата част.

Създаване на прозорец и неговите компоненти

Windows или Windows, известни също като рамки, са екземпляри от класа wx.Frames и ще бъдат съдържащата база на нашия интерфейс, нека видим един много основен примерен код, който можем да използваме като база, за да разберем работата на това, което наистина прави инструментариумът.

Импортиране на приложението :

w = gw.getWindowsWithTitle(window\_name)[0]  
w.activate()

Стартираме кода, като импортираме wx, който е нашият инструментариум, след това правим два екземпляра един за приложението и друг за рамката, която ще бъде контейнерът, след което извикваме метода Show (), който ще покаже прозореца и приложението ще бъде вътре; Разбира се всичко това е празно, тъй като не сме направили нищо, но когато стартираме нашата програма, ще видим следното:

Както виждаме, това е доста просто и дава различен аспект на това, което сме свикнали да виждаме в Python с командната конзола.

Това е всичко за тази първа част, в следващата ще задълбочим малко повече в тази тема с повече практически примери.

## Мислите ли как python ще обработва файлове?

Да вземем пример за това как нормалните хора ще се справят с файловете. Ако искаме да прочетем данните от файл или да запишем данните във файл, тогава първо ще отворим файла или ще създадем нов файл, ако файлът не съществува и след това ще изпълним нормалните операции за четене / запис , запишете файла и го затворете. По същия начин ние правим същите операции в python, използвайки някои вградени методи или функции.

# Що е алгоритъм?

Алгоритъм (от името на учения ал–Хорезми) е термин от математиката, информатиката, лингвистиката и други области, с който се описва сложно действие чрез редица от елементарни действия, които изпълняващият може да извърши в последователни стъпки без допълнителни обяснения. Обикновено изпълнението на алгоритъма включва изчисление или обработка на данни. По-строго дефинирано, алгоритъмът е ефективен метод за изчисляване на функция, който може да бъде изразен в рамките на крайно време и пространство и чрез добре дефиниран формален език. Започвайки от начално състояние и входни данни (понякога празни), инструкциите описват пресмятания, чието изпълнение преминава през краен брой добре дефинирани последователни състояния и завършва с крайно състояние, като в процеса се получават крайни резултати. Не е задължително преходът между състоянията да е детерминиран (еднозначно определен): някои алгоритми, известни като вероятностни алгоритми, съдържат елемент на случайност. Концепцията за алгоритмите съществува от векове, но частичното формулиране на понятието започва с опитите да се реши 10-ия проблем на Хилберт – „Задача за разрешимост на диофантово уравнение“, поставен от Давид Хилберт през 1900 година на Втория световен конгрес по математика в Париж. Последващите формулировки, при които се цели дефинирането на „ефективна изчислимост“или „ефективен метод“, включват рекурсивните функции на Ербран-Гьодел-Клини от 1930, 1934 и 1935 година, ламбда смятането на Алонсо Чърч от 1936 година, „Формулировка 1“ на Емил Пост от 1936 година и машината на Тюринг от 1936 – 1937 и 1939 година. Създаването на формална дефиниция на алгоритъм, съответстваща на интуитивното понятие, остава отворен въпрос и в наши дни. При все че няма общоприета формална дефиниция на алгоритъм, неформално понятието може да се определи като „набор от правила, които точно дефинират някаква поредица от операции“. Това определение обхваща всички компютърни програми, включително тези, които не извършват числени изчисления, стига те да прекратяват работа след краен брой операции. Класически пример за „алгоритъм“ е алгоритъмът на Евклид чрез изваждане за намиране на най-големия общ делител (НОД) на две цели числа, по-големи от 1. При него се изпълнява следната поредица от стъпки: На стъпка i, се дели X на Y и остатъкът се означава с R. Ако R = 0, резултатът на задачата е Y. В противен случай, резултатът съвпада с НОД на числата R и Y. Такъв алгоритъм, при който за намирането на решение е необходимо да бъде решена аналогична, но „по-малка“ задача, се нарича рекурсивен. Американските учени Джордж Булос и Ричард Джефри дават следното неформално определение за алгоритъм в своя класически учебник по математическа логика:

Никое човешко същество не може да пише достатъчно бързо или достатъчно дълго, или достатъчно дребно, за да изброи всички елементи на дадено изброимо безкрайно множество, изписвайки техните имена, едно след друго, с една и съща нотация. Но хората могат да направят нещо също толкова полезно в случая с някои изброими безкрайни множества – те могат да дадат експлицитни указания за определяне на n-тия елемент на множеството за произволно крайно n. Такива указания могат да бъдат дадени съвсем експлицитно, във форма, в която могат да бъдат изпълнени от изчислителна машина или от човек, който може да извършва само съвсем елементарни операции със символи. Изброимо безкрайно множество е такова множество, за чиито елементи може да се дефинира биективно съответствие с множеството на естествените числа. Така според Булос и Джефри алгоритъмът съдържа инструкции за процес, който „създава“ резултатни цели числа от произволно входно цяло число (или входни числа), което на теория може да бъде произволно голямо. По този начин алгоритъмът може да бъде алгебрично уравнение, като y = m + n – две произволни „входни променливи“ m и n, които създават резултат y. В същото време опитите на различни автори да дефинират понятието показват, че под „алгоритъм“ се разбира много повече от това:

Точни указания (на език, разбираем за „компютъра“) за бърз, ефективен, „добър“ процес, който специфицира „движенията“ на „компютъра“ (машина или човек, оборудван с необходимата вътрешна информация и способности), за да намери, декодира и обработи произволни входни числа/символи m е n, символи + и = ... и „ефективно“ да произведе, за „разумно“ време, краен резултат y на определено място и в определен формат. Концепцията за „алгоритъм“ се използва и за дефиниране на понятието решимост, което играе централна роля за обяснението как формалните системи възникват от малък набор аксиоми и правила. В логиката времето, необходимо за изпълнение на даден алгоритъм, не може да се измери, тъй като то не е свързано с обичайното физично измерение. Подобни несигурности, които характеризират текущите изследвания в тази област, водят до липсата на дефиниция за „алгоритъм“, която да съответства както на конкретната, така и на абстрактната употреба на термина. Алгоритмите са от съществено значение за начините, по които компютрите обработват информацията. Много компютърни програми съдържат алгоритми, които определят специфични инструкции, които компютърът трябва да изпълни в строго определен ред, за да се реши дадена задача. В този смисъл, алгоритъмът може да се разглежда като произволна поредица от операции, които могат да се симулират от пълна по Тюринг система. Към този възглед се придържат автори като Мински, Савидж и Гуревич:

Но ние също така ще твърдим, като Тюринг..., че всяка процедура, която може „естествено“ да бъде наречена ефективна, всъщност може да бъде реализирана от (проста) машина. Макар че това може да изглежда крайно, аргументите... в полза на това е трудно да се отхвърлят. Неформалният аргумент на Тюринг в полза на неговата теза доказва и по-силно твърдение: всеки алгоритъм може да бъде симулиран от машина на Тюринг... според Савидж [1987], алгоритъм е изчислителен процес, дефиниран от машина на Тюринг. Обичайно, когато един алгоритъм е свързан с обработка на информация, данните се четат от входа, изпращат се на изхода и/или се съхраняват за по-нататъшна обработка. Съхранените данни се разглеждат като част от вътрешното състояние на обекта, изпълняващ алгоритъма. На практика, състоянието се пази в една или повече структури от данни.

За всеки подобен изчислителен процес трябва строго да се дефинира един алгоритъм, като се определи така, че да е приложим при всички възможни обстоятелства, които могат да възникнат. Това ще рече, че систематично трябва да се разгледат всички условни разклонения, случай по случай, като критериите за всеки от случаите трябва да са ясно дефинирани и изчислими.

Тъй като алгоритъмът е точно определен списък от точно определени стъпки, редът на изчислението им винаги е от критично значение за работата на алгоритъма. Обикновено се предполага, че инструкциите са изрично изброени и са описани отначало-докрай, така както се изобразява една блок-схема. Това разбиране за формализацията на алгоритъма е основано на принципите на императивното програмиране. Други алтернативни концепции за алгоритмите предоставят функционалното и логическото програмиране.

## Представяне:

Алгоритмите могат да се представят с много различни видове нотация, в това число естествени езици, псевдокод, блок-схеми или програмни езици. Описанието на алгоритми на естествен език страда от обичайната склонност на езика към многословие и многосмисленост и поради това рядко се използва за формулирането на сложни алгоритми. Псевдокодът и блок-схемите представляват структурирани начини за изразяване на алгоритми, които избягват двусмислиците на естествения език и са независими от конкретния програмен език, на който алгоритмите се реализират. Програмните езици са главно насочени към изразяването на алгоритми в изпълним от компютър вид, но често се ползват и за да онагледяват, дефинират или документират алгоритмите.

Има голямо разнообразие от начини за представяне на алгоритмите – дадена програма за машина на Тюринг може да се опише като поредица от машинни таблици, като блок-схема, като рудиментарен машинен код или асемблерен код. Представянията на алгоритмите могат да се класифицират в три нива на описание на машината на Тюринг:

### Описание от високо ниво:

„...текст за описване на алгоритъм, игнориращ подробностите на реализацията му. На това ниво няма нужда да се отбелязва как машината управлява своята памет или глава.“

### Описание на реализацията:

„...текст, използван за дефиниране на начина, по който машината на Тюринг използва своята глава, и начина, по който съхранява данните в паметта си. На това ниво не се дават подробности за състоянията или преходните функции.“

### Формално описание:

Най-подробното, „на най-ниско ниво“, задава „таблица на състоянията“ на машината на Тюринг.

# PyCharm:

PyCharm е интегрирана среда за разработка (IDE), използвана за програмиране в Python . Той осигурява анализ на кода, графичен дебъгер, интегриран модулен тестер, интеграция със системи за контрол на версиите и поддържа уеб разработка с Django . PyCharm е разработен от чешката компания JetBrains. Той е междуплатформен , работи на Microsoft Windows , macOS и Linux . PyCharm има професионално издание, издадено под патентован лиценз и издание на общността, пуснато под лиценза на Apache. PyCharm Community Edition е по-малко обширно от Professional Edition.

## Характеристики:

• Помощ при кодиране и анализ , с довършване на код , осветяване на синтаксис и грешки , интегриране на Linter и бързи поправки

• Навигация по проекти и кодове: специализирани изгледи на проекти, изгледи на файлова структура и бързо прескачане между файлове, класове , методи и употреби

• Рефакторинг на Python код : включително преименуване, извличане на метод, въвеждане на променлива, въвеждане на константа, изтегляне нагоре, натискане надолу и други

• Поддръжка за уеб рамки : Django, web2py и Flask

• Интегриран дебъгер на Python

• Интегрирано тестване на единици , с покритие ред по ред

• Разработка на Python на Google App Engine

• Интегриране на контрола на версиите: унифициран потребителски интерфейс за Mercurial , Git , Subversion , Perforce и CVS със списъци с промени и сливане

• Интегриране на научни инструменти: интегрира се с IPython Notebook, има интерактивна Python конзола и поддържа Anaconda, както и множество научни пакети, включително Matplotlib и NumPy.

### ****Pycharm: среда за разработка на Python:****

PyCharm е IDE е интегрирана среда за разработка (IDE), използвана за програмиране в Python. Той осигурява анализ на кода, графичен дебъгер, интегриран модулен тестер, интеграция със системи за контрол на версиите и поддържа уеб разработка с Django. PyCharm е разработен от чешката компания JetBrains. Той е междуплатформен, работи на Microsoft Windows, macOS и Linux. PyCharm има професионално издание, издадено под патентован лиценз и издание на общността, пуснато под лиценза на Apache. PyCharm Community Edition е по-малко обширно от Professional Edition.

### ****Python рефакторинг:****

Включва преименуване, метод на извличане, въвеждане на променлива, въвеждане на константа, издърпване, натискане надолу и други.

Поддръжка на рамки web: Django, web2py и Flask

Вграден дебъгер на Python

Интегрирано модулно тестване, с покритие код по ред

Интеграция на контрол на версиите- Унифициран потребителски интерфейс за Mercurial, Git, Subversion, Perforce и CVS със списъци за промяна и сливане.

## Как да инсталираме PyCharm на Linux?

В случай на Ubuntu и деривати PyCharm е достъпен за изтегляне от софтуерния център на Ubuntu в три издания: Pro версия, EDU версия и CE версия. Всичко, което трябва да направите, е да потърсите Pycharm и той ще се появи. За останалите дистрибуции имаме по-обща инсталация, просто трябва да изтеглите файла .tar.gz от официалния сайт на Jet Brains.

След като изтеглянето приключи, отворете терминал в папката за изтегляне, където се намира файлът и въведете следното в терминала, за да го извлечете:

### Първоначална настройка на Pycharm:

Първото изпълнение на Pycharm ще ви позволи да го конфигурирате въз основа на вашите предпочитания, включително конфигуриране на вашите теми, местоположението на вашите проекти и конфигуриране на приставките, които искате да включите. Те трябва да прочетат „Споразумението за политика за поверителност“ и да го приемат, за да продължат. Направете това, можете да конфигурирате предпочитаната от вас тема по-долу, като цяло има три теми на потребителския интерфейс: Intellij, Darcula и GTK +. Те могат да улеснят стартирането на IDE, използвайки скриптове за стартиране, но можете да го пропуснете.

След това те могат да конфигурират приставките, които искате да включите във вашата инсталация. Началният екран на приставката ще се появи, както следва:

В края на конфигурацията, ще се отвори прозорец, подобен на следния за да създадете нов проект, да го отворите или да излезете от системата за контрол на версиите.

След като са избрали проект, началният екран на приложението, който ще видите, ще бъде следният:

# Видео файлове**:**

Видео файлът съхранява визуална, звукова и друга информация. Категорията Видео файлове съдържа голямо разнообразие от видео формати, които използват различни кодеци, за да кодират и компресират различните типове видео. Атрибутите на видео сигнала включват пиксели (pixel dimensions), аудио канали (audio channels), кадри в секунда (frame rate) и други. Често срещани видео файлови разширения са: .MPG, .MOV, .WMV, .AVI и .RM

Когато се работи с видео файлове е нужно голямо количество данни, които да изобразят сигнала по подходящ начин. Тази информация се компресира и записва във файл контейнер.

Съществува голямо разнообразие от формати за запазване на видео. Някои от тях са подходящи за записване на видео, други на редактирането му, а трети предоставят самото видео.

## **Анатомия на видео файла:**

Видео файлът съдържа контейнер, видео и аудио сигнали и кодеци. Контейнер файл се използва за да идентифицира различни видове данни. Различните формати контейнери съдържат множество видео и аудио потоци, субтитри, информация за данни и метаданни (тагове), както и информация за синхронизация, нужда за да работи всичко едновременно. Примери за контейнери са AVI и Quicktime MOV, MPEG и други. Кодек е софтуер, който се използва за кодиране и декодиране на видео сигнал. Видео програмите използват кодеци за да записват и четат файлове.

### Разделителна способност:

Преди цифровата ера, хоризонталната резолюция се е измервала във вертикални линии и е представлявала броя на снимките във видео изображение, зависещи от честотната характеристика на записващото устройство.

Всеки цифров видео сигнал се характеризира с резолюция, измервана в пиксели. Стандартна разделителна способност на аналоговия сигнал е 720 × 576 пиксела за Европейския стандарт на разлагане 625/50 (PAL и SECAM)) с честота на кадрите от 50 Hz (2 × 25) и 720 × 480 пиксела при Американски стандарт на разширяване 525 / 60 (NTSC) с честота – 59,94 Hz (2 × 29,97).). В израза 720 × 480, първото число е броят на точките в един ред (хоризонтална резолюция), а второто число броят на активните линии в изображението (разделителна способност по вертикала).

Стандарта цифрова телевизия HDTV – High Definition (English високо качество) включва резолюция до 1920 × 1080 при честота от 50 Hz (60 Hz за САЩ). Това означава 1920 пиксела на линия и 1080 линии.

При триизмерната телевизия, резолюцията се измерва във воксел (англ. Voxel) – пиксели, представляващи точки (кубчета) в тримерното пространство. Пример за резолюция на 3D видео е 512 × 512 × 512.

## Какво е Gizeh

Gizeh е библиотека за рисуване на вектори в Python. Той имплементира някои класове върху cairocffi, за да го направи по-интуитивен.

### Какво ни трябва?

Изисква се Python 3 и pipenv. Всички модули могат да бъдат инсталирани от PyPI (Python Package Index) с помощта на pip.

Съотношението ширина към височина на картината (англ. aspect ratio) – основен параметър за всеки видеозапис. В края на XIX век немите филми и впоследствие филмите от стандартния <<класическия>> период, имали съотношение на страните на картината 4:3 (в кинематографията се записва 1,33:1). Считало се, че екран с такова съотношение на страните е най-близък до зрителното поле на човека. Появата на телевизията скоро след това възприела това съотношение и почти всяка телесистема (следователно и телевизор) имали съотношение 4:3. Първите компютърни монитори също унаследили този стандарт. Обаче, в кинематографията вече в началото на 50-те години на XX век с появата на панорамното (широкоекранно) кино представите за идеалния екран се поразлюляли. Широкоекранните кинематографични системи приели съотношението 2,75:1, стремейки се към максимален ефект на присъствието, така, че да се направят границите на картината незабележими. Главната причина била в това, че полето на бинукулярното зрение на човека се приближава към съотношение 2:1. За да се доближи формата на картината към естественото зрително поле (и следователно да се усили възприятието от филма) се разработили киносистеми с панорамна картина. Излъчването на широкоекранни филми по телевизията налагало или да се изреже картината чрез пансканиране или чрез добавяне на празни полета отдолу и отгоре на картината чрез (англ. Letterbox). И двата способа водели към загуба на части от изображението или на неговите качества. Към днешен ден класическия формат 1,37:1 не се използва въобще в кинематографията, отстъпвайки своето място на формата 1,85:1. Заради това, при избора на формат на екрана за телевизия с висока резолюция (HDTV, HD) бил одобрен стандарта 16:9 (1,78:1), по-близък до разпространените кино формати. Цифровата телевизия със стандартна разделителна способност в основата си също е ориентирана към 16:9, прилагайки цифрово анаморфироване. Всичко това, по замисъла на създателите си е направено с цел да гмурне зрителя по-дълбоко в атмосферата на филма. Има и алтернативно обяснение за прехода към широкоформатна картина: възможност за отдаване на зали под наем, първоначално непригодни за кино, стремейки се към влошаване качеството на пиратските видеоклипове и телевизионни копия.

### Честота на кадрите:

Броя на неподвижни изображения, които се появяват едно след друго в рамките на една секунда,за да създадат движение на обектите на екрана се нарича честота на кадрите. Колкото е по-висока скоростта на кадрите, толкова по-плавно и естествено изглежда движението. Минималната скорост, с която човек възприема изображението гладко е 16 кадъра в секунда (в действителност това число е индивидуално за всеки човек). В киното се използва стандартно 24 кадъра в секунда. Системите за телевизия PAL и SECAM използват 25 кадъра (25 fps или 25 Hz), а NTSC – 30. Компютърните видео файлове с добро качество като правило използват 30 Hz. Горната гранична честота на трептенията, възприемана от човешкия мозък, е индивидуална за всеки, зависи от условията на наблюдаване и има средна стойност 39 – 42 Hz. Някои съвременни професионални камери снимат с честота до 120 Hz. Специални камери, с които се изследва детайлно полет на куршум или взрив, снимат със 1000 Hz. Съществуват и свръхскоростни камери, които снимат с няколко милиона кадъра в секунда. При тях фотографският филм е неподвижно закрепен по вътрешната повърхност на барабан, а изображенията се получават от въртяща призма.

Съществува и безкадрово видео. Принципът на работа се заключава в следното: светлочувствителен сензор предава с висока честота своето моментно състояние, което се записва на носител. Тук няма отделни кадри, а само масив от информация за изменението във времето на сигнала от всеки датчик (пиксел). При възпроизвеждането също няма кадри – на екрана пикселите менят своя цвят в съответствие със записаните данни. Ако пикселът не мени цвета си, то той и не се обновява. За най-добро гледане на такова видео трябва специален монитор.

### Скорост:

Количеството обработваема информация за единица време се нарича скорост на видеопотока, побитова скорост (също ширина) и се измерва в бит/с – бит в секунда или Мбит/с – мегабит в секунда; на английски съответно bit/s и Mbit/s. Колкото е по-добро качеството му, толкова по-голяма побитова скорост е необходима. Например за формата VideoCD ширината на видео потока е 1 Mbit/s, за DVD тя е около 5 Mbit/s, а форматът за цифрова телевизия HDTV използва ширина около 10 Mbit/s. С помощта на понятието за скорост на видео потока е много удобно да се оценява качеството на видеото при неговото предаване през интернет. Различават се два вида управление на ширината на потока във видеокодеците – постоянен битрейт (англ. constant bit rate, CBR) и променлив битрейт (англ. variable bit rate VBR). Концепцията на VBR, макар и не много популярна е да съхрани максимално качеството на видеото, докато намалява сумарно обема на предавания видео поток. В този случай при сцени с много и бързи движения на картината, ширината на видео потока расте, а на забавените сцени, където картината се променя бавно, спада. Това е много удобно при буферирано предаване на видеото и при предаване по компютърна мрежа. Но при безбуферни системи на предаване и в директен ефир (например при телеконференция) това не е подходящо – в тези случаи трябва да се използва постоянна скорост на видео потока. Скоростта за конвертиране на звука от аналогов в дигитален файл се нарича скорост на звука. Файл форматът е структура за пакетиране на данни. Почти всеки компютърен файл съдържа повече от един тип данни. Видео файловете са сложни като цяло, но всички те трябва да бъдат запазени в един от основните видове контейнери. Между всеки един вид контейнер (като Quicktime или AVI) има огромно разнообразие от атрибути на видео сигнала, използвани кодеци или компресия. Файлове с еднакви контейнери също могат да бъдат много различни помежду си. Например ако имаме два AVI файла, те могат да бъдат с еднакво разширение, но това не означава, че могат да се възпроизведат от един и същ софтуер. Възможно е да са с различна резолюция или честота на кадрите, която да не може да се изпълни на всеки софтуер. Разширенията на контейнерите включват: MOV, AVI, FLV, MP4 и MXF.

### Кодеци:

Всеки видео файл контейнер съдържа видео и аудио данни. Тези данни са създадени от софтуер наречен кодек (накратко от компресор или декомпресор). Кодеците могат да бъдат възприемани като помощници на приложението, което програмата или операционната система използва, за да възпроизведе видео файл. Без подходящ кодек, видео файлът не може да бъде стартиран.

Понякога видео кодеците са патентовани и могат да включват такси за лиценз, но повечето софтуери, които възпроизвеждат видео файлове, както и операционната система, имат вградени кодеци и също позволяват инсталацията на други, допълнителни кодеци.

Разликата между файл формат контейнера и кодека е, че контейнерът е начин за съхраняване на данни, а кодекът е метод да кодира и декодира видео и аудио сигнал в този контейнер.

Например, възможно е да се вземе MTS файл, създаден с H.264 кодек и да се превърне в MOV файл като просто се „преопакова“ (англ. rewrapping). По време на процеса „преопаковане“ (англ. rewrap), се променят само няколко бита в началото и края на файла, а останалите остават абсолютно същите.

Съществува огромно разнообразие от кодеци, което прави съвместимостта на видео файловете много сложна. Кодекът не може да се разпознае от разширението на файла, в който е използван, а софтуерът дава само частична информация за него. Единствено софтуер за редакция на видео може да разпознае какъв кодек се използва в даден файл.

Един от най-използваните кодеци през XXI век е H.264 (наричан още MPEG-4 част 10 и AVC). Той предлага перфектна компресия с високо качество и гъвкавост. Използван при висока побитова скорост, той осигурява високо качество, което може да се забележи например при Blu-ray диск.

## **Необработени видео формати:**

Некомпресирано видео, наричано още Чист HDMI е цифрова видео информация, която не е била компресирана или не е била обработвана с компресиране, когато видеото е било записвано директно от видео камерата. То представлява потока от данни или файловия формат използван от цифровите видео камери. Целта е значително по-високо качество, в сравнение с конвертирането със загуба на данни, позволявайки дори повишаване на резолюцията. HDMI спецификацията се отнася за няколко вида некомпресирано цифрово видео. Въпреки че често HD камерите включват HDMI интерфейс за възпроизвеждане или дори преглед на живо, видео процесорът на камерите, използван за некомпресирано видео трябва да бъде в състояние да възпроизведе пълната резолюция на образа, при посочената честота на кадрите в реално време, без да пропуска кадър. Затова използването на некомпресирано видео през HDMI, често се нарича „Чист HDMI“ Некомпресирано видео се поддържа от Никон камери DSLR с Expeed 3 (FR) видео процесор, Canon EOS-1D C, Canon 5D Mark III и професионалните видео камери, които използват съответните некомпресирани видео формати, като CinemaDNG или ArriRaw. Некомпресираното видео доставя по-висока резолюция на образа и острота, отколкото съответстващия некомпресиран формат за снимки. Предимствата на високото качество са, че няма замъгляване на при движение или влошаване поради компресиране. Понастоящем няма стандартизиран видео файлов формат, с изключение на HDMI, който използва YCbCr и RGB форматите. Това създава необходимостта файловете да се запазват с описание на използваната резолюция и видео настройка.

## Компресирани видео формати:

Компресиран видео формат е спецификация, за цифрово представяне на видео под формата на файл или поток от информация. Примери за компресирани видео формати са MPEG-2 Part 2, MPEG-4 Part 2, H.264 (MPEG-4 Part 10), Theora, Dirac, RealVideo RV40, и VP8. Специфична софтуерна или хардуерна имплементация на видео компресиране и/или декомпресиране с използването на специфичен видео компресиращ формат се нарича видео кодек; пример за видео кодек е Xvid, който имплементира кодиране и декодиране на видео, използвайки MPEG-2 Part 2 компресиран видео формат.

Видео, кодирано в съответствие с видео компресионен формат обикновено се комбинира с аудио поток (кодиран със съответния компресиран аудио формат), в мултимедиен контейнер с формат като AVI, MP4, FLV, RealMedia или Matroska. В този случай, обикновено потребителят не използва H.264 файл, а mp4 файл, който представлява MP4 контейнер, съдържащ H.264-кодирано видео, обикновено заедно с AAC-кодирано аудио. Форматът на мултимедийния контейнер може да съдържа един от многото различни видео компресионни формати; например MP4 контейнер може да съдържа видео или в MPEG-2 Part 2 или H.264 видео компресионен формат. Друг пример е файл от типа WebM, който е контейнер на форматът Matroska, който използва видео компресионен формат VP8 и аудио компресионен формат Vorbis, въпреки че форматът на контейнера Matroska, е в състояние да съдържа и други видео компресионни формати.

* Импортиране на библиотеките:

import moviepy.editor as mpy

import gizeh as gz

from math import pi

В тази част от кода ние казваме на Python да добави библиотеките, които ще бъдат използвани в този урок, за да бъде прост. Добавяме всички инструменти на MoviePy в неговия редакторски модул и го преименуваме на mpy, за да бъде кратък. В случая на Gizeh, ние го добавяме напълно и го преименуваме на gz. И накрая импортираме pi от модула math python, за да улесним някои изчисления в кода.

BLUE = (59/255, 89/255, 152,255)

def render\_text(t):

surface = gz.Surface(640, 60, bg\_color=(1, 1, 1))

Функцията render\_text ще настрои всичко, от което Gizeh се нуждае, за да нарисува това, което искаме. Тази функция е важна, за да направи процеса лесен за разбиране, като отделя кода на Gizeh от кода на Moviepy. Тази функция получава като параметър времето t, което ще бъде обяснено по-късно. Може би сте забелязали, че BLUE е RGB цвят. Gizeh работи с RGB, но по скала от 0 до 1, така че бялото е (1, 1, 1) еквивалентно на (255, 255, 255). В случая на BLUE ние изрично разделяме синьото на Stack Builders (59, 89, 152), за да съответства на формата на Gizeh. Гизе обича да работи като художник; има нужда от платно, за да начертае всички неща, които сме посочили в нашия код. Така че трябва да създадем повърхност с ширина, височина и цвят на фона. В този случай нашата повърхност ще бъде ширина=640, височина=60 и цвят на фона = (1, 1, 1).

text = gz.text(“Let’s build together”, together”, fontfamily=”Charter”,

fontside=30, fontweight=’bold’, fill=BLUE, xy=(320, 40))

С готовата повърхност можем да започнем да рисуваме неща в Gizeh. За да начертаем текста в нашата повърхност, ще използваме gz.text. Тази функция получава като първи параметър текста за изобразяване. Другите параметри се използват за стилизиране на текста (ключови параметри, които се обясняват сами по себе си). xy получава координати за позициониране на текста вътре в повърхността. В този случай xy = (320, 40).

text.draw(surface)

return surface.get\_npimage()

Сега с готовия текст можем да го начертаем в повърхността, създадена по-рано. За да постигнем това, просто трябва да кажем на текста върху коя повърхност трябва да бъде начертан. За да получим повърхността с цялото съдържание, начертано върху нея, използваме surface.get\_npimage(). Това връща масив numpy с всички чертежи върху него. Този масив може да се използва с Moviepy.

text = mpy.VideoClip(render\_text, duration=10)

Време е да използвате Moviepy! Сега, използвайки VideoClip, можем да създадем екземпляр на видео в паметта. С функцията render\_text, която връща текста в масив numpy, можем да конвертираме масива в клип на Moviepy. Функцията mpy.VideoClip получава като задължителни параметри: какво искаме да изобразим или кадъра, който ще бъде преобразуван във видео клип и продължителността, която трябва да има. Помните ли параметъра t във функцията render\_text? Вътрешно mpy.VideoClip запазва кадъра на клипа, съответстващ на времето t в кадъра. Полученият видеоклип автоматично ще се оразмери, за да съответства на предишното ни платно (640x60), така че няма нужда да го указвате.

## Добавяне на файл с изображение**:**

Сега, след като имаме готов текст, можем да добавим някои изображения към видеоклипа, за да го направим по-привлекателен за зрителите. За целта ще използваме следния код:

SB\_logo\_path = ‘./static/StackBuildersLogo.jpg’

sb\_logo = mpy.ImageClip(SB\_logo\_path).\

set\_position((‘center’, 0)).\

resize(width=200)

Функцията mpy.ImageClip получава като параметър изображението за изобразяване. В този случай пътят на изображението се задава в променливата SB\_logo\_path. Използвайки атрибутите на ImageClip, е възможно да позиционирате изображението с помощта на координати x, y. В този случай е възможно да се смесват ключови думи като център, дъно и горе със специфични координати. И накрая, атрибутът за преоразмеряване намалява размера на изображението, запазвайки съотношението му, използвайки ключовите аргументи ширина или височина, както във фрагмента. Възможно е изрично да зададете персонализиран размер, като използвате кортеж с новите размери (ширина, височина).

# Използвани библиотеки за създаване на програмата:

## **CV2**

OpenCV е библиотека от програмни функции главно за компютърно зрение в реално време. Първоначално разработен от Intel , по-късно се поддържа от Willow Garage , след това от Itseez. Библиотеката е междуплатформена и е лицензирана като безплатен софтуер с отворен код под Apache License 2. От 2011 г. OpenCV включва GPU ускорение за операции в реално време. OpenCV е написан на езика за програмиране C++ , както и основният му интерфейс, но все още запазва по-малко изчерпателен, но обширен по-стар C интерфейс. Всички по-нови разработки и алгоритми се появяват в C++ интерфейса. Има езикови обвързвания в Python , Java и MATLAB / Octave . Интерфейсът за приложно програмиране ( API ) за тези интерфейси може да бъде намерен в онлайн документацията. Разработени са библиотеки за обвивки на няколко езика, за да се насърчи приемането от по-широка аудитория. Във версия 3.4, JavaScriptо обвързвания за избрано подмножество от функции на OpenCV бяха пуснати като OpenCV.js, за да се използват за уеб платформи. Ако библиотеката открие интегрираните примитиви за производителност на Intel в системата, тя ще използва тези патентовани оптимизирани процедури, за да се ускори. От септември 2010 г. се разработва базиран на Compute Unified Device Architecture ( CUDA ) интерфейс за графичен процесор (GPU). Базиран на OpenCL GPU интерфейс се разработва от октомври 2012 г., документацията за версия 2.4.13.3 може да бъде намерена на docs.opencv.org. OpenCV работи на настолни операционни системи : Windows , Linux , macOS , FreeBSD , NetBSD и OpenBSD , както и на мобилни операционни системи: Android , iOS , Maemo , BlackBerry 10 и QNX . Потребителят може да получи официални версии от SourceForge или да вземе най-новите източници от GitHub.

## **Pyautogui**

• PyAutoGUI позволява на вашите Python скриптове да контролират мишката и клавиатурата, за да автоматизират взаимодействията с други приложения. API е проектиран да бъде прост. PyAutoGUI работи на Windows, macOS и Linux и работи на Python 2 и 3. PyAutoGUI има няколко функции:

• Преместване на мишката и щракване в прозорците на други приложения.

• Изпращане на натискания на клавиши към приложения (например за попълване на формуляри).

• Направете екранни снимки и дадено изображение (например на бутон или квадратче за отметка) и го намерете на екрана.

• Намерете прозореца на приложението и го преместете, преоразмерете, увеличете, минимизирайте или затворете (само за Windows, в момента).

• Показване на кутии за предупреждение и съобщения.

* Примери:

>>> import pyautogui

>>> screenWidth, screenHeight = pyautogui.size() # Get the size of the primary monitor.

>>> screenWidth, screenHeight

(2560, 1440)

>>> currentMouseX, currentMouseY = pyautogui.position() # Get the XY position of the mouse.

>>> currentMouseX, currentMouseY

(1314, 345)

>>> pyautogui.moveTo(100, 150) # Move the mouse to XY coordinates.

>>> pyautogui.click() # Click the mouse.

>>> pyautogui.click(100, 200) # Move the mouse to XY coordinates and click it.

>>> pyautogui.click('button.png') # Find where button.png appears on the screen and click it.

>>> pyautogui.move(400, 0) # Move the mouse 400 pixels to the right of its current position.

>>> pyautogui.doubleClick() # Double click the mouse.

>>> pyautogui.moveTo(500, 500, duration=2, tween=pyautogui.easeInOutQuad) # Use tweening/easing function to move mouse over 2 seconds.

>>> pyautogui.write('Hello world!', interval=0.25) # type with quarter-second pause in between each key

>>> pyautogui.press('esc') # Press the Esc key. All key names are in pyautogui.KEY\_NAMES

>>> with pyautogui.hold('shift'): # Press the Shift key down and hold it.

pyautogui.press(['left', 'left', 'left', 'left']) # Press the left arrow key 4 times.

>>> # Shift key is released automatically.

>>> pyautogui.hotkey('ctrl', 'c') # Press the Ctrl-C hotkey combination.

>>> pyautogui.alert('This is the message to display.') # Make an alert box appear and pause the program.

Този пример плъзга мишката в квадратна спирална форма в MS Paint (или друга програма за рисуване на графики):

>>> distance = 200

>>> while distance > 0:

pyautogui.drag(distance, 0, duration=0.5) # move right

distance -= 5

pyautogui.drag(0, distance, duration=0.5) # move down

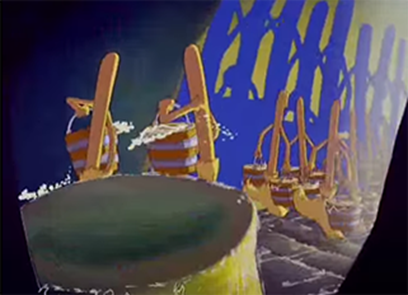
pyautogui.drag(-distance, 0, duration=0.5) # move left

distance -= 5

pyautogui.drag(0, -distance, duration=0.5) # move up

Ползата от използването на PyAutoGUI, за разлика от скрипт, който директно генерира файла с изображение, е, че можете да използвате инструментите за четки, които MS Paint предоставя.

## **Сейфове:**

Подобно на омагьосаните метли от Sorcerer's Apprentice, програмирани да продължават да пълнят (и след това да препълват) ваната с вода, грешка във вашата програма може да я накара да излезе извън контрол. Трудно е да използвате мишката, за да затворите програма, ако курсорът на мишката се движи сам. Като функция за безопасност по подразбиране е активирана функция за безопасност при отказ. Когато се извика функция на PyAutoGUI, ако мишката е в някой от четирите ъгъла на основния монитор, те ще повишат pyautogui.FailSafeException. Има забавяне от една десета от секундата след извикване на всяка функция на PyAutoGUI, за да се даде време на потребителя да удари мишката в ъгъла, за да задейства защитата при отказ. Можете да деактивирате тази защита от грешки, като зададете . СИЛНО ПРЕПОРЪЧВАМ ДА НЕ ДЕАКТИВИРАТЕ FAILSAFE.pyautogui.FAILSAFE = False. Закъснението от десета секунда се задава от pyautogui.PAUSEнастройката, която е 0.1 по подразбиране. Можете да промените тази стойност. Има и pyautogui. DARWIN\_CATCH\_UP\_TIME настройка, която добавя допълнително забавяне на macOS след събития с клавиатура и мишка, тъй като изглежда, че операционната система се нуждае от забавяне, след като PyAutoGUI издаде тези събития. Зададено е 0.01по подразбиране, добавяйки допълнително забавяне от стотна секунда.

13. Кодът на Python за запис във видео файл:

**import** cv2  
**import** numpy **as** np  
**import** pyautogui  
**import** pygetwindow **as** gw  
  
#window\_name = 'notepad'  
window\_name = **'word'**#fourcc = cv2.VideoWriter\_fourcc(\*"XVID")  
#fourcc = cv2.VideoWriter\_fourcc(\*"DIVX")  
#fourcc = cv2.VideoWriter\_fourcc(\*"MJPG")  
fps = 30.0  
record\_seconds = 10  
  
w = gw.getWindowsWithTitle(window\_name)[0]  
w.activate()  
  
#out = cv2.VideoWriter(r"demo\output.avi", fourcc, fps, tuple(w.size))  
out = cv2.VideoWriter(**r"demo\output.avi"**,cv2.VideoWriter\_fourcc(**'M'**,**'J'**,**'P'**,**'G'**), fps, tuple(w.size))  
  
**for** i **in** range(int(record\_seconds \* fps)):  
 img = pyautogui.screenshot(region=(w.left, w.top, w.width, w.height))  
 frame = np.array(img)  
 frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
 out.write(frame)  
 cv2.imshow(**"screenshot"**, frame)  
 **if** cv2.waitKey(1) & 0xFF == ord(**'q'**):  
 **break**out.release()  
cv2.destroyAllWindows()

1. Cv2 е името за импортиране на модул за opencv-python, „Неофициални предварително изградени OpenCV пакети само за CPU за Python“. Традиционният OpenCV има много сложни стъпки, включващи изграждане на модула от нулата.
2. Импортираната numpy част от кода казва на Python да пренесе библиотеката NumPy в текущата среда. След това частта as np от кода казва на Python да даде на NumPy псевдонима np.
3. PyAutoGUI позволява на нашите Python скриптове да контролират мишката и клавиатурата, за да автоматизират взаимодействията с други приложения. Междуплатформен модул за получаване на GUI информация за контролиране на прозорците на приложението.
4. Както NumPy дава псевдонима np, така и pygetwindow го съкращаваме като gw.
5. На 6 и 7 ред, казваме на програмата, че създаваме файл с име „ notepad ” или „ word “.
6. FPS е съкращение от Frames per second. Това е единица, която измерва производителността на дисплея на устройството при заснемане и възпроизвеждане на видео. FPS се използва за измерване на скоростта на кадрите - броят изображения, показвани последователно всяка секунда.
7. Record seconds е времето в секунди на продължителността на записа. В моят случай, това време е 10 секунди.
8. Следващия ред е цикъл, на който казваме колко време да записва.
9. Въвеждаме променлива, която активира прозорец, в който трябва да запишем видеото.
10. Създаваме обект за запис на видеото.
11. Създаваме цикъл for, който ще работи толкова, колкото сме оказали в променливата.
12. Записването на видеото се получава с помощта на запазването на скрийншот.
13. Скрийншотите се запазват в променливата img.
14. Преобразуваме пикселите в pyautogui.
15. Преобразуваме цветовете от формат gtp в RGB.
16. Създаваме рамка и я извеждаме.
17. И добавяме условие, че при натискане на Q, записа спира.

# Списък на използваната литература:

1. https://softwareacademy.bg/index.php?q=info&info=django+now/

2. https://softuni.bg/blog/advantages-of-django-framework

3. https://www.pro-soft.bg/web-saitove/django/

4. https://bg.wikipedia.org/wiki/REST

5. https://www.python.org/

6. https://chibisov.github.io/drf-extensions/docs/

7. https://bg.altlibs.net/lib/django-drf-filepond

8. https://softuni.bg/blog/what-is-html5-and-how-it-works

9. https://html.start.bg/%D0%9A%D0%B0%D0%BA%D0%B2%D0%BE+%D0%B5+HTML%3F-19832

10. https://www.sourcetrail.com/bg/python/django/solved-django-serializer-method-field-read-write/

11. http://javac.bg/?p=522

12. https://www.django-rest-framework.org/api-guide/serializers/