

Методические указания для выполнения лабораторных работ

Программирование на Python

Инструментом программиста является компьютер, рассмотрим его устройство. Все вычисления в компьютере производятся центральным процессором. Файлы с программами хранятся в постоянной памяти (на жестком диске), а в момент выполнения загружаются во временную (оперативную) память. Ввод информации в компьютер осуществляется с помощью клавиатуры (устройства ввода), а вывод – с помощью монитора (устройства вывода).



Компьютер способен работать только с двумя видами сигналов: 1 или 0 (машинным кодом).

Писать программы вида 1010101010010101010 для человека сложно, мышление его устроено иначе, поэтому появились программы-трансляторы с



языка программирования, понятного человеку, на машинный язык, понятный компьютеру. Языки программирования, которые приближены к машинному уровню, называют языками низкого уровня (например, язык ассемблера). Другой вид языков – языки высокого уровня (например, Python, Java, C#), еще больше приближенные к мышлению человека.







Как правило, языки программирования создавались под конкретную задачу. На сегодняшний день существует множество различных языков программирования, наиболее популярные из которых представлены в таблице.

Началом общения с компьютером послужил машинный код. Затем в 50-ые годы двадцатого века появился низкоуровневый язык ассемблера, наиболее приближенный к машинному уровню. Он привязан к процессору, поэтому его изучение равносильно изучению архитектуры процессора. На языке ассемблера пишут программы и сегодня, он незаменим в случае небольших устройств (микроконтроллеров), обладающих очень ограниченными ресурсами памяти.

Следующий этап (80-ые годы) характеризуется появлением объектно-ориентированного программирования (ООП), которое должно было упростить создание крупных промышленных программ. Появляется ученый – Б. Страуструп, которому недостаточно было возможностей языка C, поэтому он расширяет этот язык путем добавления ООП. Новый язык получил название C++.

В 90-ые годы появляются персональные компьютеры и сеть Интернет, потому требуются новые технологии и языки программирования. Язык Java создавался с оглядкой на C++ и с перспективой развития сети Интернет.

Примерно в одно время с Java появляется Python. Разработчик языка – математик Гвидо ван Россум занимался долгое время разработкой языка ABC, предназначенного для обучения программированию. С ростом сети Интернет потребовалось создавать динамические сайты – появился серверный язык программирования PHP.

		Язык ассемблера	Микрокомпьютеры с очень ограниченными ресурсами.		50-ые
		Fortran	Математические расчеты.		
		Basic	Языки для обучения программированию.		60-70-ые
		Pascal			
		C	Системное программирование (драйвера и пр.)		ОС UNIX
		C++	Включает все возможности языка C, реализует ООП подход.	Потребность в больших программах - появился подход ООП.	80-ые
		Java	Крупные программы для бизнеса (ООП). Сложно написать плохую программу.	Потребность в программистах и переносимости. Автоматизировать кофемашину.	90-ые Персональные ПК, Интернет
		Python	Автоматизация рутинной деятельности (быстро), обучение программированию.		
		PHP	Разработка динамических сайтов.		
		C# (.NET)	Крупные программы для бизнеса (ООП). Много общего с Java. Зависимость от продуктов Microsoft.	Обобщение и объединение: собрать всё лучшее, что было до этого.	2000-ые

В 2000-ые годы наблюдается тенденция объединения технологий вокруг крупных корпораций. В это время получает развитие язык C# на платформе .NET.

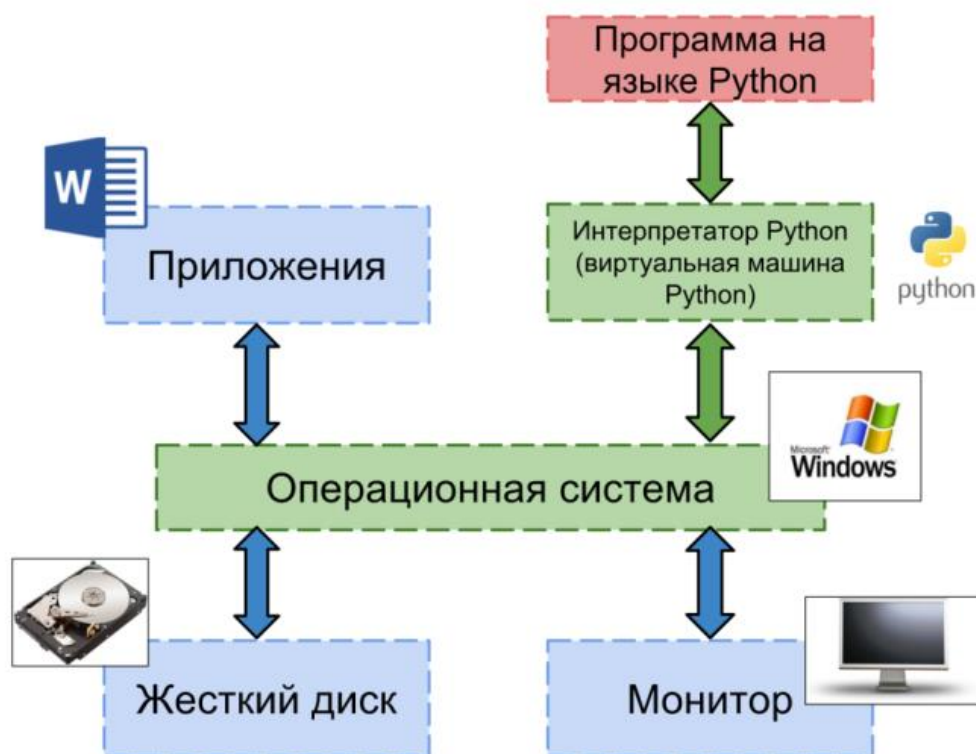
Язык программирования Python

Питон является одним из наиболее широко используемых языков программирования в следующих областях:

1. Системное программирование.

2. Разработка программ с графическим интерфейсом.
3. Разработка динамических веб-сайтов.
4. Интеграция компонентов.
5. Разработка программ для работы с базами данных.
6. Быстрое создание прототипов.
7. Разработка программ для анализа данных.
8. Разработка программ для научных вычислений.
9. Разработка игр.

Выполнение программ осуществляется операционной системой (Windows, Linux и пр.). В задачи операционной системы входит распределение ресурсов (оперативной памяти и пр.) для программы, запрет или разрешение на доступ к устройствам ввода/вывода и т. д.



Для запуска программ на языке Python необходима программа-интерпретатор (виртуальная машина) Python. Данная программа скрывает от Python-программиста все особенности операционной системы, поэтому, написав программу на Python в системе Windows, ее можно запустить, например, в GNU/Linux и получить такой же результат.

Начало работы с Google Colab

Лабораторные работы будут осуществляться в лаборатории Гугл (Google Colaboratory), в нее можно перейти по следующей ссылке:

<https://colab.research.google.com/notebooks/welcome> (рис.1).

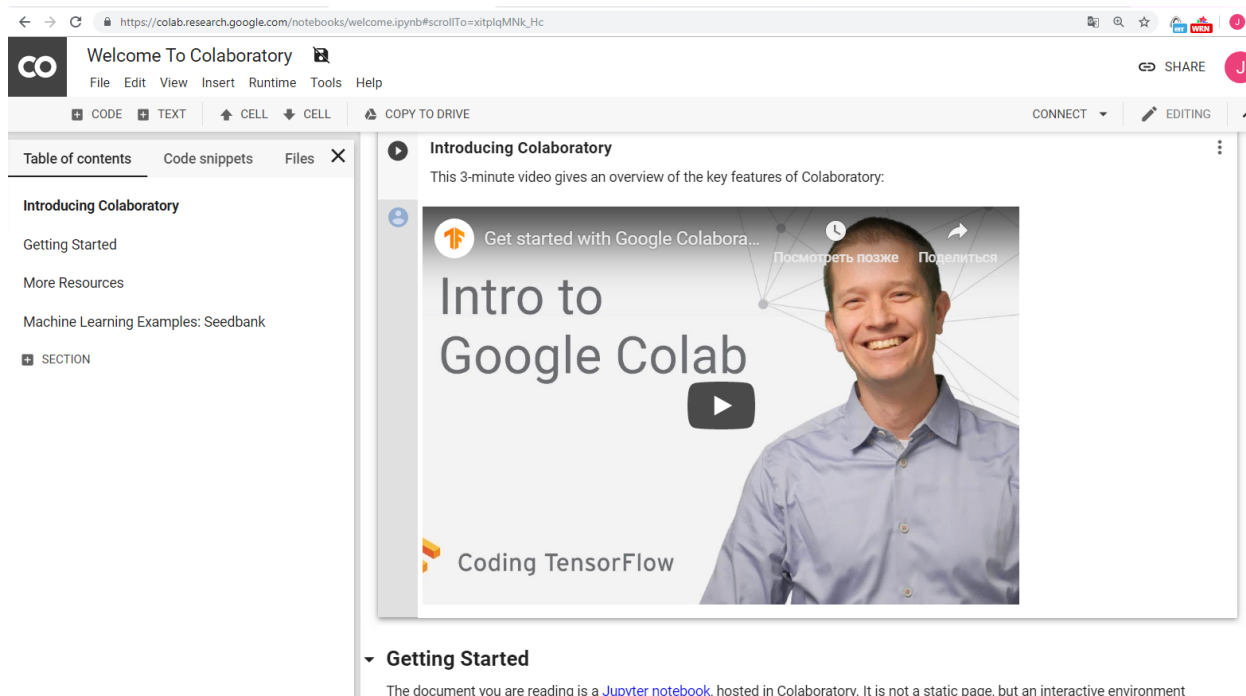


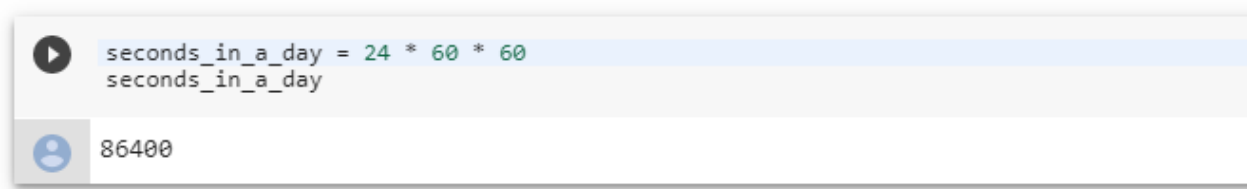
Рис. 1. Начальная страница лаборатории Гугл

Работа будет вестись в блокноте Jupyter (<https://jupyter.org/>), размещенным в лаборатории. Блокнот Jupyter представляет собой не статическую страницу, а интерактивную среду, которая позволяет писать и выполнять код на Python и других языках.

Отличительной особенностью написания кода в блокноте Jupyter является то, что код разбивается на ячейки (cell), каждая из которых может быть выполнена отдельно.

Файлы, создаваемые и редактируемые в блокноте Jupyter имеют расширение **.ipynb**.

Для начала работы можно запустить код, представленный на первой странице лаборатории (рис.2). Для этого нужно нажать на стрелку в соответствующей ячейке или выделить ячейку и нажать **ctrl+enter**. Для выполнения кода последовательно во всех ячейках можно воспользоваться сочетанием клавиш **ctrl+F9**.



```
seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

86400

Рис. 2. Пример кода на языке Python в блокноте Jupyter

В данной программе производится подсчет количества секунд в сутках. Переменной **seconds_in_a_day** присваивается значение, равное $24(\text{часа}) \cdot 60(\text{минут}) \cdot 60(\text{секунд})$. Вторая строка предназначена для вывода значения, хранящегося в переменной **seconds_in_a_day**.

Для создания нового файла вашей программы нужно перейти в меню File->New Python 3 notebook.

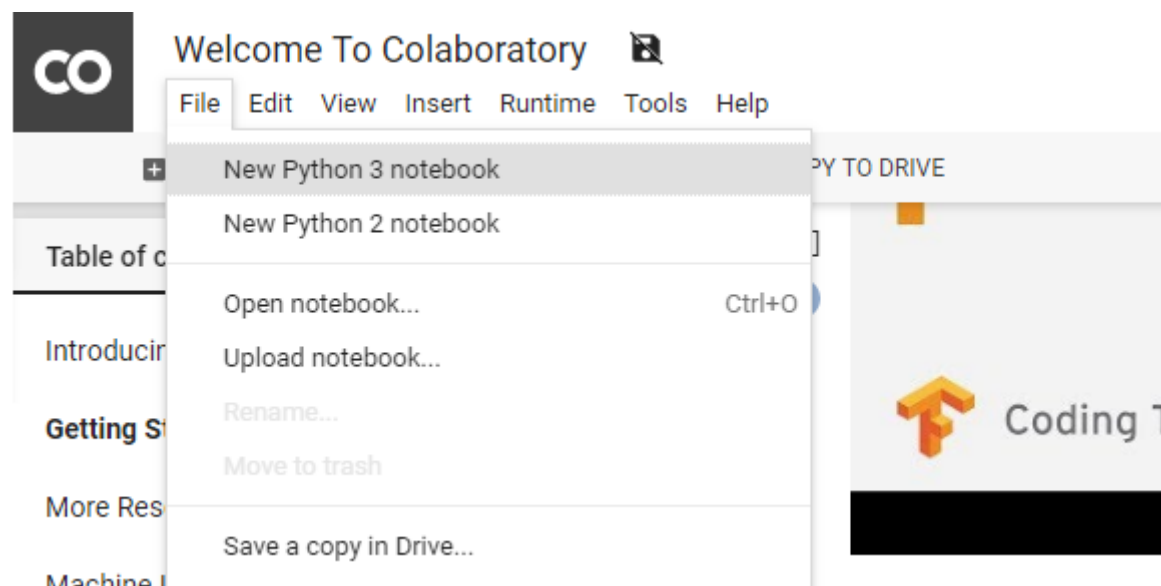


Рис. 3. Создание нового программного файла .ipynb

В новой вкладке откроется файл Untitled2.ipynb (рис. 4), с которым вы будете работать далее.

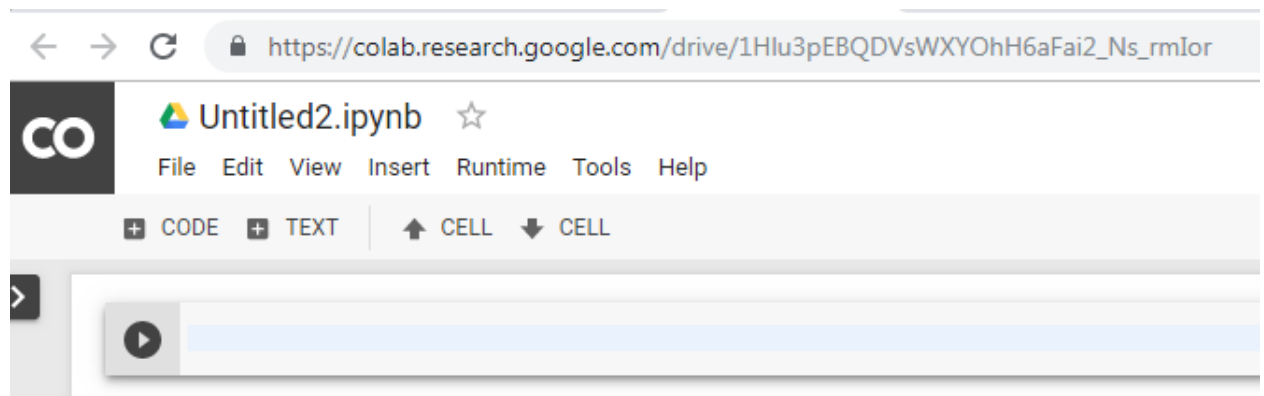
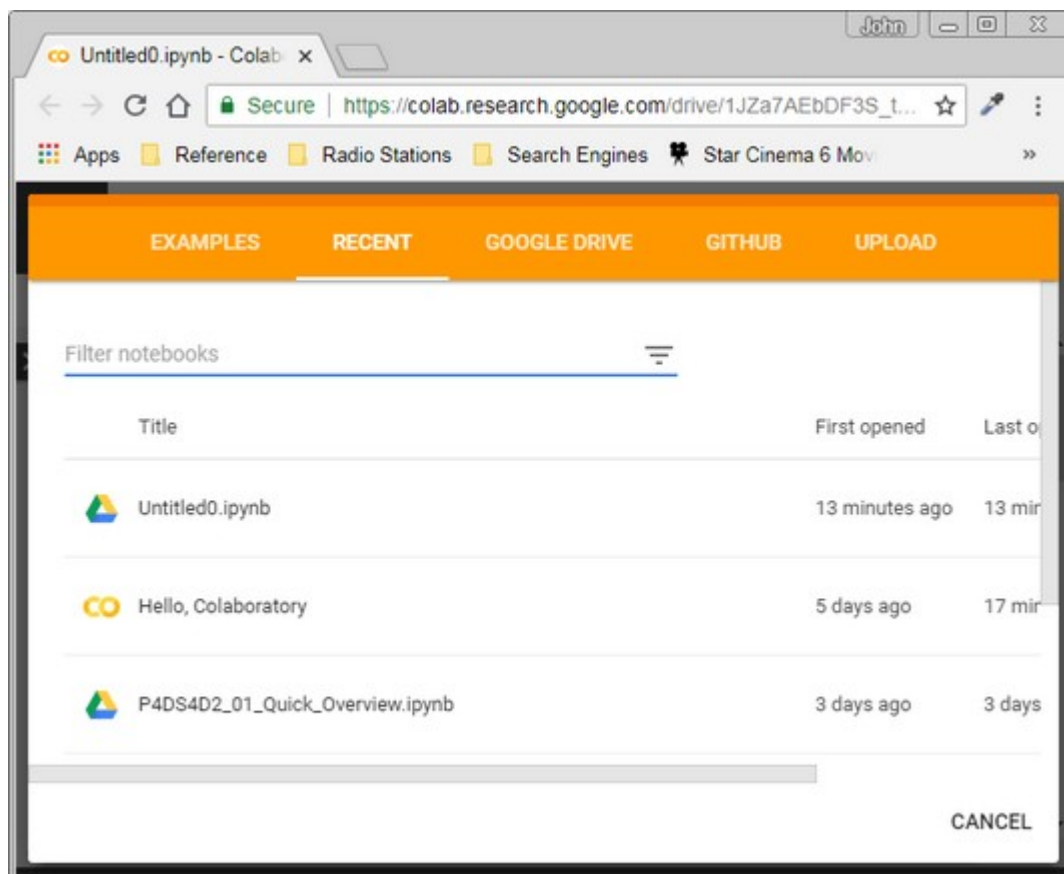


Рис. 4. Пример нового программного файла

Переименовать файл можно выбрав в меню File → Rename.

Открытие ранее созданных проектов осуществляется выбором File → Open Notebook. Появится диалоговое окно, отображающее недавно открытые файлы.



Для того чтобы сохранить файл с новым именем на жесткий диск необходимо в меню выбрать File->download .ipynb. После выполнения лабораторной работы нужно будет сохранять файлы на диске.

Для добавления новой ячейки выберите Insert->Code cell (рис.5).

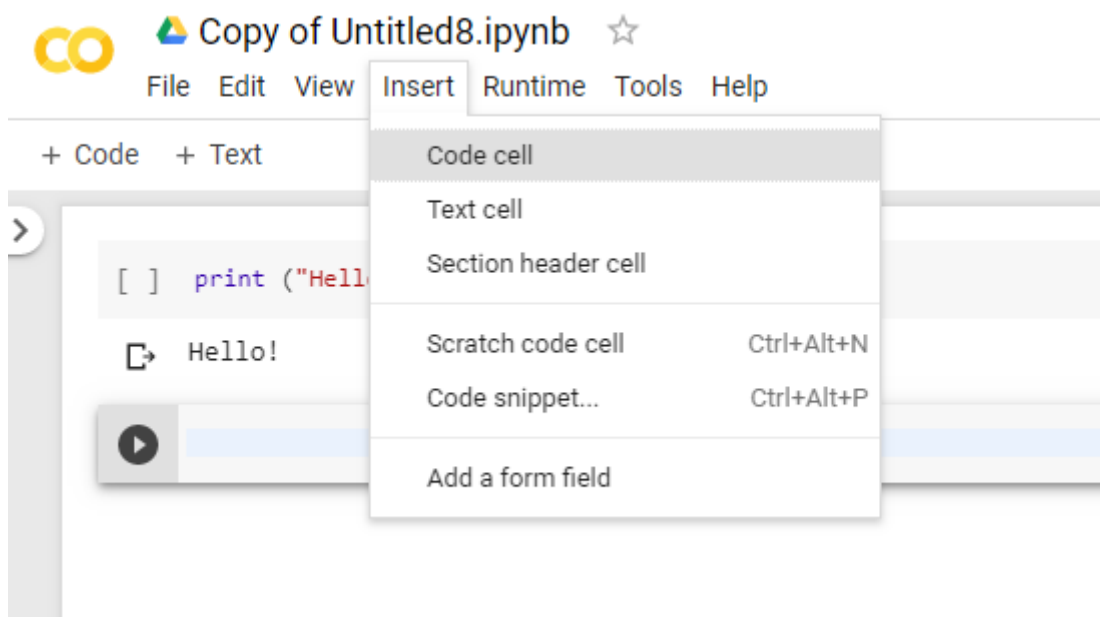


Рис. 5. Добавление новой ячейки кода

При наведении курсора мыши на ячейку кода, справа сверху появляется всплывающее меню (рис. 6). Нажатие на значок «сообщение» позволяет добавить комментарий к ячейке. Нажатие на корзину – удалить ячейку. Нажатие на «шестиренку» даст возможность редактирования настроек данной ячейки.

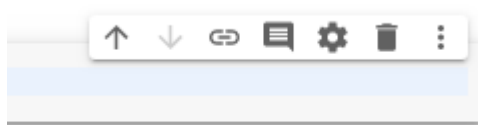


Рис. 6. Всплывающее окно ячейки кода

Для ознакомления с функциями можно воспользоваться документацией по языку Python: <https://docs.python.org/3/tutorial/>

Задания к лабораторным работам

Лабораторная работа №1. Введение в Python

Задание 1

Напишите программу для решения примера (по вариантам).

Предусмотрите проверку деления на ноль. Все необходимые переменные пользователь вводит через консоль. Запись `|пример|` означает «взять по модулю», т.е. если значение получится отрицательным, необходимо сменить знак с минуса на плюс.

Для вычисления примеров вам понадобится библиотека `math`. Подключить ее можно, записав в ячейке кода: `import math`.

Вариант 1.

$$k = \ln \left| \left(y - \sqrt{|x|} \right) \cdot \left(x - \frac{y}{z + x^2 / 4} \right) \right|$$

Вариант 2.

$$d = \frac{2 \cos(x - \pi / 6)}{1/2 + \sin^2(y)} + \frac{|y - x|}{3}$$

Вариант 3.

$$w = \frac{(x / y) \cdot (z + x) \cdot e^{|x-y|} + \ln(1 + e)}{\sin^2(y) - (\sin(x) \cdot \sin(y))^2}$$

Вариант 4.

$$b = \frac{3 + e^{y-1}}{1 + x^2 \cdot |y - \operatorname{tg}(z)|}$$

Задание 2

Разработать программу для вычисления выражения и вывода полученного результата. Соответствующие исходные данные ввести с клавиатуры.

Вариант 1:

$$p = \begin{cases} \sqrt{|a \cdot b|} + 2 \cdot c, a \cdot b < -2 \\ a^3 + b^2 - c^2, -2 \leq a \cdot b \leq 2 \\ a^c - b, a \cdot b > 2 \end{cases}$$

Исходные данные: a,b,c.

Вариант 2:

$$h = \begin{cases} \arctg(x + |y|), x < y \\ \arctg(|x| + y), x > y \\ (x + y)^2, x = y \end{cases}$$

Исходные данные: x,y.

Вариант 3:

$$b = \begin{cases} \ln(x / y) + (x^2 + y)^3, x / y > 0 \\ \ln |x / y| + (x^2 + y)^3, x / y < 0 \\ (x^2 + y)^3, y \neq 0, x = 0 \\ 0, y = 0 \end{cases}$$

Исходные данные: x,y.

Вариант 4:

$$b = \begin{cases} \sin(x + y) + 2 \cdot (x + y)^2, x - y > 0 \\ \sin(x - y) + (x - y)^3, x - y < 0 \\ |x^2 + \sqrt{y}|, y \neq 0, x = 0 \\ 0, y = 0 \end{cases}$$

Исходные данные: x,y.

Лабораторная работа №2. Работа с файлами. списки

Задание. Программа должна создавать файл *.xls, записать в него сгенерированный случайным образом массив чисел. Затем, с помощью реализованного алгоритма сортировки, одного из предложенных преподавателем, записать отсортированную последовательность чисел в ранее созданный файл *.xls.

Алгоритмы сортировки:

- Сортировка выбором
- Сортировка вставками
- Сортировка “Методом пузырька”
- Сортировка Шелла
- Быстрая сортировка

Процесс создание, сохранения и работы с файлами на диске

1. Способ. Простая загрузка файла на локальный диск

```
from google.colab import files // из библиотеки google.colab загружается
                                раздел files
with open('example.txt', 'w') as f: // открытие файла «example.txt» с правом
                                    чтения в переменную f
    f.write('some content') // запись в файл f текста «some content»
files.download('example.txt') // сохранение файла на диске как «example.txt»
```

2. Способ. Для того чтобы подгрузить файл в colab вам потребуется обратиться к своему google диску. В приведенном ниже примере показано, как подключить диск Google Drive во время выполнения с помощью кода авторизации и как записывать и читать файлы там. После выполнения вы увидите новый файл (foo.txt) по адресу <https://drive.google.com/>.

После монтирование google-диска необходимо перейти по предложенной ссылке URL (рис. 7).

```
[ ] from google.colab import drive
drive.mount('/content/drive')
```


 Go to this URL in a browser: [https://accounts.google.c](https://accounts.google.com/authorize?client_id=...)
Enter your authorization code:
.....
Mounted at /content/drive

Рис. 7. Процесс подключение google-диска

Самая нижняя строка на рис. 7. Оповещает о том, что ваш google-диск подключен.

Файлы вашего google-диска находятся по адресу: «/content/drive/My Drive/»

Для просмотра существующих документов на вашем google-диске воспользуемся командой `!ls` (рис. 8),

```
[113] !ls /content/drive/My\ Drive
```



 'Colab Notebooks' foo.txt 'Диссер-Выделение лиц.СНС.pdf'
file1.txt timeline

Рис. 8. Просмотр существующих документов на google-диске

Создадим файл `foo.txt` (рис. 9), записав в нем фразу «Hello Google Drive!».



```
with open('/content/drive/My Drive/foo.txt', 'w') as f:
    f.write('Hello Google Drive!')
!cat /content/drive/My\ Drive/foo.txt
```


 Hello Google Drive!

Рис. 9. Изменение содержимого файла `foo.txt` и его вывод

Последняя строка кода на рис. 9. Позволяет вывести содержимое файла «foo.txt», где

`!cat` – команда вывода содержимого файла,

`/content/drive/My\ Drive/` – путь к файлу,

`foo.txt` – название файла.

Для дополнительной информации можно обратиться:
<https://colab.research.google.com/notebooks/io.ipynb#scrollTo=7taylj9wpsA2>

Лабораторная работа №3.

Вариант 1.

Напишите программу по следующему описанию. Есть класс "Воин". От него создаются два экземпляра-юнита. Каждому устанавливается здоровье в 100 очков. В случайном порядке они бьют друг друга. Тот, кто бьет, здоровья не теряет. У того, кого бьют, оно уменьшается на 20 очков от одного удара. После каждого удара надо выводить сообщение, какой юнит атаковал, и сколько у противника осталось здоровья. Как только у кого-то заканчивается ресурс здоровья, программа завершается сообщением о том, кто одержал победу.

Вариант 2.

Напишите программу по следующему описанию:

Есть класс Person, конструктор которого принимает три параметра (не учитывая self) – имя, фамилию и квалификацию специалиста. Квалификация имеет значение заданное по умолчанию, равное единице.

У класса Person есть метод, который возвращает строку, включающую в себя всю информацию о сотруднике.

Класс Person содержит деструктор, который выводит на экран фразу "До свидания, мистер ..." (вместо троеточия должны выводиться имя и фамилия объекта).

В основной ветке программы создайте три объекта класса Person. Посмотрите информацию о сотрудниках и увольте самое слабое звено.

В конце программы добавьте функцию input(), чтобы скрипт не завершился сам, пока не будет нажат Enter. Иначе вы сразу увидите, как удаляются все объекты при завершении работы программы.

Вариант 3.

Разработайте программу по следующему описанию.

В некой игре-стратегии есть солдаты и герои. У всех есть свойство, содержащее уникальный номер объекта, и свойство, в котором хранится принадлежность команде. У солдат есть метод "иду за героем", который в качестве аргумента принимает объект типа "герой". У героев есть метод увеличения собственного уровня.

В основной ветке программы создается по одному герою для каждой команды. В цикле генерируются объекты-солдаты. Их принадлежность команде определяется случайно. Солдаты разных команд добавляются в разные списки.

Измеряется длина списков солдат противоборствующих команд и выводится на экран. У героя, принадлежащего команде с более длинным списком, поднимается уровень.

Отправьте одного из солдат первого героя следовать за ним. Выведите на экран идентификационные номера этих двух юнитов.

Вариант 4.

В качестве практической работы попробуйте самостоятельно перегрузить оператор сложения. Для его перегрузки используется метод `__add__()`. Он вызывается, когда объекты класса, имеющего данный метод, фигурируют в операции сложения, причем с левой стороны. Это значит, что в выражении `a + b` у объекта должен быть метод `__add__()`. Объект `b` может быть чем угодно, но чаще всего он бывает объектом того же класса. Объект `b` будет автоматически передаваться в метод `__add__()` в качестве второго аргумента (первый – `self`).

Отметим, в Python также есть правосторонний метод перегрузки сложения – `__radd__()`.

Согласно полиморфизму ООП, возвращать метод `__add__()` может что угодно. Может вообще ничего не возвращать, а "молча" вносить изменения в какие-то уже существующие объекты. Допустим, в вашей программе метод перегрузки сложения будет возвращать новый объект того же класса.

Лабораторная работа №4

Создать программу для работы с базой данных, при этом добавив обработку исключительных ситуаций. База данных должна содержать не менее 5 связанных таблиц. У программы должен быть графический интерфейс [<https://colab.research.google.com/notebooks/forms.ipynb>,]. По сохраненным данным в таблицах построить минимум 3 различных графика [<https://colab.research.google.com/notebooks/charts.ipynb>, <https://colab.research.google.com/notebooks/widgets.ipynb>] с помощью библиотеки matplotlib.

Создание и редактирование форм

Формы позволяют удобно редактировать код. Выделив текущую ячейку кода необходимо выбрать раздел **Insert** → **Add form field**. При изменении значения в форме изменяется соответствующая строка кода.

Для сокрытия кода необходимо выбрать раздел **Edit** → **Show/hide code**.