

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

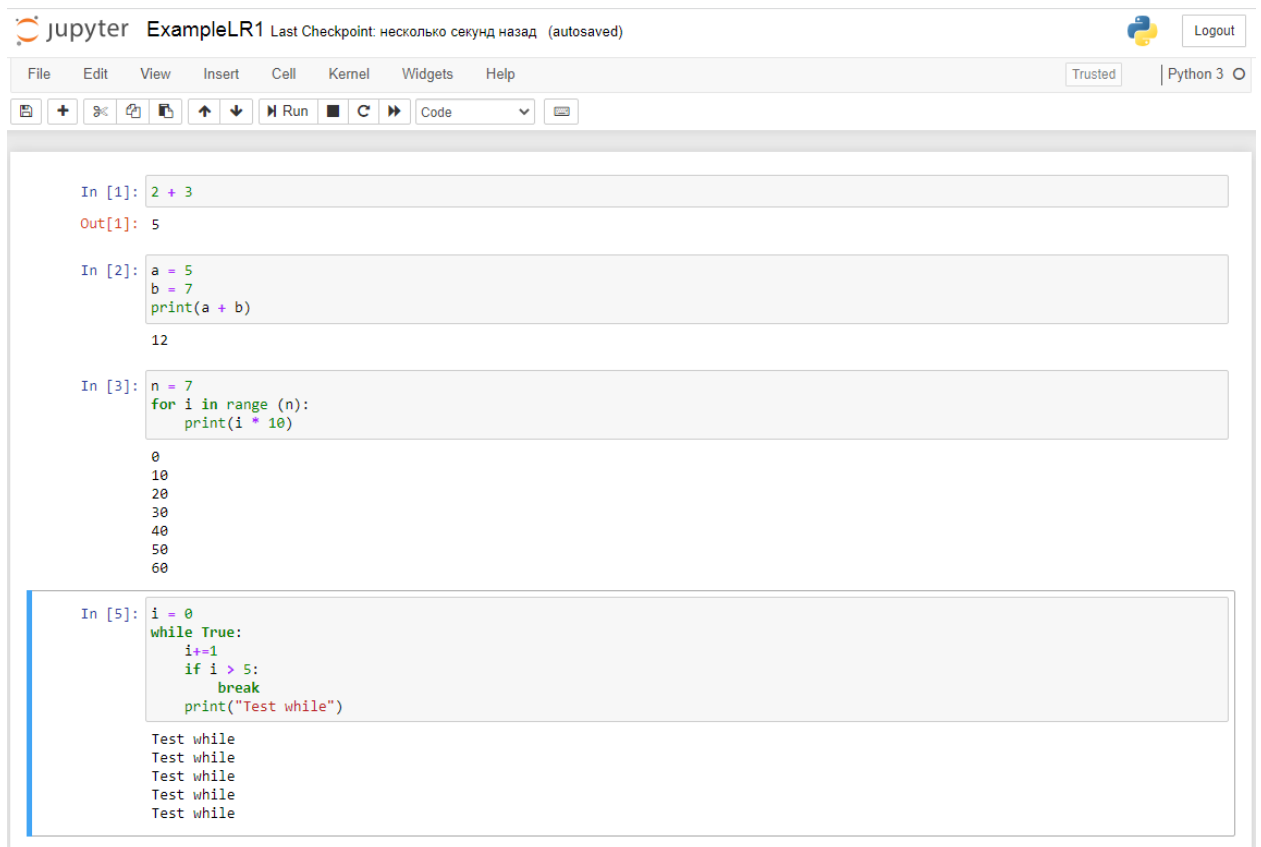
**Отчет о лабораторной работе №1 по дисциплине технологии
распознавания образов**

Выполнил:
Выходцев Егор Дмитриевич,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г

1. Примеры из методических указаний



The screenshot shows a Jupyter Notebook window titled "ExampleLR1" with a status bar indicating "Last Checkpoint: несколько секунд назад (autosaved)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The notebook contains five code cells:

- Cell 1:** `In [1]: 2 + 3` with output `Out[1]: 5`.
- Cell 2:** `In [2]: a = 5, b = 7, print(a + b)` with output `12`.
- Cell 3:** `In [3]: n = 7, for i in range(n): print(i * 10)` with output `0, 10, 20, 30, 40, 50, 60`.
- Cell 4:** `In [5]: i = 0, while True: i+=1, if i > 5: break, print("Test while")` with output `Test while` repeated five times.

Рисунок 1 – Работа с простыми примерами Jupyter Notebook

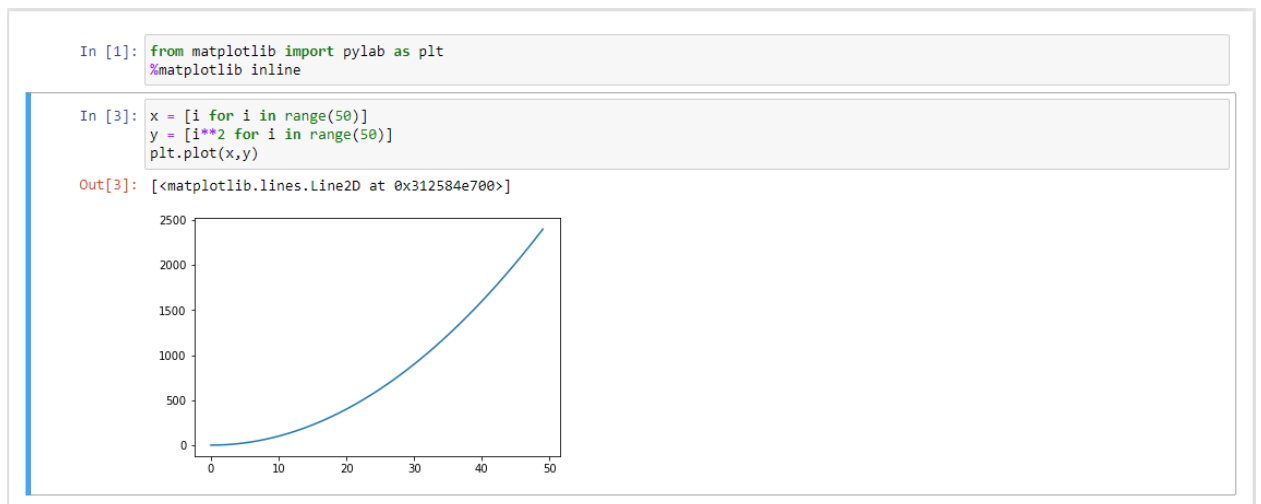


Рисунок 2 – Рисование графиков

2. Задания преподавателя

#1 Счастливый билетик (рис. 3)

```

Output: 103

In [7]: ticket_number = int(input("Enter your ticket"))
        Enter your ticket123042

In [8]: sum1 = ticket_number // 100000 + ticket_number // 10000 % 10 + ticket_number // 1000 % 10

In [9]: print(sum1)
        6

In [10]: sum2 = ticket_number % 1000 // 100 + ticket_number % 100 // 10 + ticket_number % 10

In [11]: print(sum2)
        6

In [12]: if sum1 == sum2:
        print("Yes")
        else:
        print("No")
        Yes

```

Рисунок 1 – Код программы и результат выполнения для числа «123042»

#2 Пароль (рис 2,3).

```

Ввод [5]: password = input("Enter your pasword")
        name = input("What is your name?")
        Enter your paswordAandrei123
        What is your name?Andrei

Ввод [7]: if password.isalpha() or password.isdigit():
        print("weak")
        exit(-1)
        if password.islower() or password.isupper():
        print("weak")
        exit(-1)

Ввод [9]: unic = set(password)
        if len(unic) < 4:
        print("weak")
        exit(-1)

Ввод [10]: if name.lower() in password.lower():
        print("weak")
        exit(-1)
        else:
        print("strong")
        weak

```

Рисунок 2 – Код программы и результат её выполнения

```
Ввод [11]: password = input("Enter your pasword")
name = input("What is your name?")
```

```
Enter your paswordan12dRei
What is your name?Andrei
```

```
Ввод [12]: if password.isalpha() or password.isdigit():
            print("weak")
            exit(-1)
            if password.islower() or password.isupper():
                print("weak")
                exit(-1)
```

```
Ввод [13]: unic = set(password)
            if len(unic) < 4:
                print("weak")
                exit(-1)
```

```
Ввод [14]: if name.lower() in password.lower():
            print("weak")
            exit(-1)
            else:
                print("strong")
```

```
strong
```

Рисунок 3 – Вывод программы для другого пароля и такого же имени

#3 Числа Фибоначчи (рис 4).

```
Ввод [20]: amount = int(input("Enter the amount "))
```

```
Enter the amount 10
```

```
Ввод [21]: a, b = 0, 1
            for i in range(amount):
                sum = a + b
                a = b
                b = sum
                print(sum)
```

```
1
2
3
5
8
13
21
34
55
89
```

Рисунок 4 – Код программ и результат её выполнения для числа 10

#4 Время исследований (рис. 5-7).

Для анализа выбран dataset, анализирующий смертность от загрязнения воздуха в разных странах, в строка кода ниже представлены необходимые доп. модули для работы с данными, а также открытие файла .csv и считывание данных с него. После прохождения по файлу формируются два списка: Общее количество смертей от загрязнения воздуха в целом и количество смертей от загрязнения воздуха озоном за период 1990 - 2017г.

```
Ввод [1]: import csv
from math import sqrt

with open('dth_to_pol.csv', 'r', newline='') as csvfile:
    data = csv.reader(csvfile, delimiter=',')
    afg_total_pol = []
    afg_ozone_pol = []
    for row in data:
        if row[0] == "Afghanistan":
            afg_total_pol.append(float(row[3]))
            afg_ozone_pol.append(float(row[6]))
```

Затем нам необходимо посчитать среднее значение коэффициентов из списка

```
Ввод [3]: mean_total = sum(afg_total_pol) / len(afg_total_pol)
mean_ozone = sum(afg_ozone_pol) / len(afg_ozone_pol)
print("Среднее значение коэффициента загрязнения воздуха в Афганистане с "
      f"1990 по 2017 год: {mean_total}")
)
print("Среднее значение коэффициента загрязнения воздуха озоном в "
      "Афганистане с "
      f"1990 по 2017 год: {mean_ozone}")
)
```

Среднее значение коэффициента загрязнения воздуха в Афганистане с 1990 по 2017 год: 252.84255072458163

Среднее значение коэффициента загрязнения воздуха озоном в Афганистане с 1990 по 2017 год: 5.797505605371072

Для того, чтобы посчитать стандартное отклонение выбранных величин создадим две промежуточные переменные, которые будут содержать в себе сумму квадратов разности для каждого элемента из списка, поделённую на количество элементов.

```
Ввод [5]: var1 = sum((elem-mean_total)**2 for elem in afg_total_pol) / len(afg_total_pol)
st_dev_total = sqrt(var1)
var2 = sum((elem-mean_ozone)**2 for elem in afg_ozone_pol) / len(afg_ozone_pol)
st_dev_ozone = sqrt(var2)
print("Стандартное отклонение коэффициента общего загрязнения воздуха: "
      f"{st_dev_total}")
)
print("Стандартное отклонение коэффициента загрязнения воздуха озоном: "
```

Рисунок 5 – Код программы

```
Ввод [5]: var1 = sum((elem-mean_total)**2 for elem in afg_total_pol) / len(afg_total_pol)
st_dev_total = sqrt(var1)
var2 = sum((elem-mean_ozone)**2 for elem in afg_ozone_pol) / len(afg_ozone_pol)
st_dev_ozone = sqrt(var2)
print("Стандартное отклонение коэффициента общего загрязнения воздуха: "
      f"{st_dev_total}")
)
print("Стандартное отклонение коэффициента загрязнения воздуха озоном: "
      f"{st_dev_ozone}")
)
```

Стандартное отклонение коэффициента общего загрязнения воздуха: 37.35656399089619

Стандартное отклонение коэффициента загрязнения воздуха озоном: 0.09717587256065482

Для того, чтобы найти коэффициенты уравнения линейной зависимости посчитает сумму произведений элементов из двух списков, а также сумму квадратов элементов из списка коэффициентов общего загрязнения воздуха. Затем посчитает по формуле коэффициент k и b. После выводим их на экран, а также значения функции для каждой точки списка коэффициентов общего загрязнения воздуха.

```
Ввод [6]: sum_ab = 0
sum_square = 0
for idx, elem in enumerate(afg_total_pol):
    sum_ab += elem * afg_ozone_pol[idx]
    sum_square += elem**2
size = len(afg_total_pol)
k_lin = (size * sum_ab - sum(afg_total_pol) * sum(afg_ozone_pol)) / (size * sum_square - sum(afg_total_pol)**2)
b_lin = mean_ozone - mean_total * k_lin
func_val = []
for elem in afg_total_pol:
    func_val.append(k_lin * elem + b_lin)
print(f"Уравнение линейной зависимости: y = {k_lin}x + {b_lin}")
print("Значения функции на кривой методом наименьших квадратов: ")
for val in func_val:
    print(val)
```

Уравнение линейной зависимости: y = -0.00048149182135542336x + 5.919247225635601

Значения функции на кривой методом наименьших квадратов:

5.77505135072677
5.778999266912121
5.7849287958966285
5.785011728466223
5.780980626729237
5.780570732469591
5.78123116360491
5.781325119009701
5.781412872176521
5.781234327095285
5.782266677711574

Рисунок 6 – Код программы, продолжение

Уравнение линейной зависимости: $y = -0.00048149182135542336x + 5.919247225635601$
Значения функции на кривой методом наименьших квадратов:

```
5.77505135072677
5.778999266912121
5.7849287958966285
5.785011728466223
5.780980626729237
5.780570732469591
5.78123116360491
5.781325119009701
5.781412872176521
5.781234327095285
5.782606457244574
5.783522615764744
5.787637650720681
5.788617094747192
5.789676076564802
5.793801344799917
5.7963822528383595
5.800549250447244
5.804888299644632
5.809102509246886
5.812584474584094
5.816047786447732
5.819391261674628
5.822512803044716
5.825145928668351
5.826896555523149
5.829367653181076
5.830680952160238
```

Для расчёта коэффициента парной корреляции разделим формулу на числитель и знаменатель. Для числителя посчитаем произведение разностей элементов каждого из списков со средним значением этих списков. В знаменателе считаем произведение средних квадратических отклонений.

```
Ввод [7]: cor_chisl = 0
for idx, elem in enumerate(afg_total_pol):
    cor_chisl += (elem - mean_total)*(afg_ozone_pol[idx] - mean_ozone)
sqr_diff_total = sum((elem-mean_total)**2 for elem in afg_total_pol)
sqr_diff_ozone = sum((elem-mean_ozone)**2 for elem in afg_ozone_pol)
r_xy = cor_chisl / sqrt(sqr_diff_total * sqr_diff_ozone)
print(f"Коэффициент парной корреляции: {r_xy}")
```

Коэффициент парной корреляции: -0.185096151561045

Ввод []: Значение -0.185 можно рассматривать как почти полное отсутствие корреляции.

Рисунок 7 – Код программы, продолжение

3. Решение вычислительной задачи (рис. 8).

Выбранная задача: Написание алгоритма решета Эратосфена.

Условие задачи - написание алгоритма решета Эратосфена для нахождения всех простых чисел до заданного числа.

1. Все четные числа, кроме двойки, - составные, т. е. не являются простыми, так как делятся не только на себя и единицу, а также еще на 2.
2. Все числа кратные трем, кроме самой тройки, - составные, так как делятся не только на самих себя и единицу, а также еще на 3. Число 4 уже выбыло из игры, так как делится на 2.
3. Число 5 простое, так как его не делит ни один простой делитель, стоящий до него.
4. Если число не делится ни на одно простое число, стоящее до него, значит оно не будет делиться ни на одно сложное число, стоящее до него. Последний пункт вытекает из того, что сложные числа всегда можно представить как произведение простых. Поэтому если одно сложное число делится на другое сложное, то первое должно делиться на делители второго. Например, 12 делится на 6, делителями которого являются 2 и 3. Число 12 делится и на 2, и на 3.

Алгоритм Эратосфена как раз заключается в последовательной проверке делимости чисел на предстоящие простые числа. Сначала берется первое простое и из ряда натуральных чисел отсеиваются все кратные ему. Затем берется следующее простое и отсеиваются все кратные ему и так далее. В конце список превращается в множество, чтобы избавиться от всех нулей, кроме первого.

```
Ввод [1]: #!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == "__main__":
    n = int(input("Enter the num to count simple numbers for: "))
    numbers = []
    for i in range(n + 1):
        numbers.append(i)
    numbers[1] = 0
    for idx, elem in enumerate(numbers):
        if idx > 1:
            if numbers[idx] != 0:
                j = idx * 2
                while j <= n:
                    numbers[j] = 0
                    j += idx
    numbers = set(numbers)
    numbers.remove(0)
    print(numbers)
```

```
Enter the num to count simple numbers for: 23
{2, 3, 5, 7, 11, 13, 17, 19, 23}
```

Рисунок 8 – Код программы и результат выполнения для числа 23

4. Ответы на контрольные вопросы

1. Как осуществляется запуск Jupyter notebook?

Jupyter Notebook входит в состав Anaconda. Для запуска Jupyter Notebook необходимо перейти в папку Scripts (она находится внутри каталога, в котором установлена Anaconda) и в командной строке набрать следующую команду: `ipython notebook`

В результате будет запущена оболочка в браузере.

2. Какие существуют типы ячеек в Jupyter notebook?

Существует два вида ячеек:

- Ячейка кода содержит код, который должен быть выполнен в ядре, и отображает его вывод ниже.
- Ячейка Markdown содержит текст, отформатированный с использованием Markdown, и отображает его вывод на месте при запуске.

3. Как осуществляется работа с ячейками в Jupyter notebook?

После выбора ячейки «Code» в неё можно записать код на языке Python и нажать Ctrl+Enter или Shift+Enter, в первом случае введенный вами код будет выполнен интерпретатором Python, во втором – будет выполнен код и создана новая ячейка, которая расположится уровнем ниже.

Alt+Enter – выполнить содержимое ячейки и вставить новую ячейку ниже.

4. Что такое "магические" команды Jupyter notebook? Какие "магические" команды Вы знаете?

Важной частью функционала Jupyter Notebook является поддержка магии. Под магией в IPython понимаются дополнительные команды, выполняемые в рамках оболочки, которые облегчают процесс разработки и расширяют ваши возможности. Список доступных магических команд можно получить с помощью команды «%lsmagic».

Для работы с переменными окружения используется команда %env.
(%env a = 5)

Запуск Python кода из “.py” файлов, а также из других ноутбуков – файлов с расширением “.ipynb”, осуществляется с помощью команды %run.
(%run ./test.py)

Для измерения времени работы кода используйте %%time и %timeit.

%%time позволяет получить информацию о времени работы кода в рамках одной ячейки.

%timeit запускает переданный ей код 100000 раз (по умолчанию) и выводит информацию среднем значении трех наиболее быстрых прогонов.

%matplotlib – используется для отображения объектов графиков на экране, ключ после него указывает каким способом отображать график. При запуске без ключей он использует сторонний backend и отображает график в

отдельном окне. При вводе команды `%matplotlib notebook` график появляется не в отдельном окне, а прямо внутри вашего блокнота, при этом он так же интерактивен. Команда `%matplotlib inline` указывает, что график необходимо построить все в той же оболочке Jupyter, но теперь он выводится как обычная картинка. Данный способ удобен тем, что позволяет проводить очень много экспериментов в рамках одного окна (точнее web-страницы). В этом статическом режиме, никакие изменения не отобразятся до тех пор пока не будет выполнена команда `plt.show()`.

5. Самостоятельно изучите работу с Jupyter notebook и IDE PyCharm и Visual Studio Code.

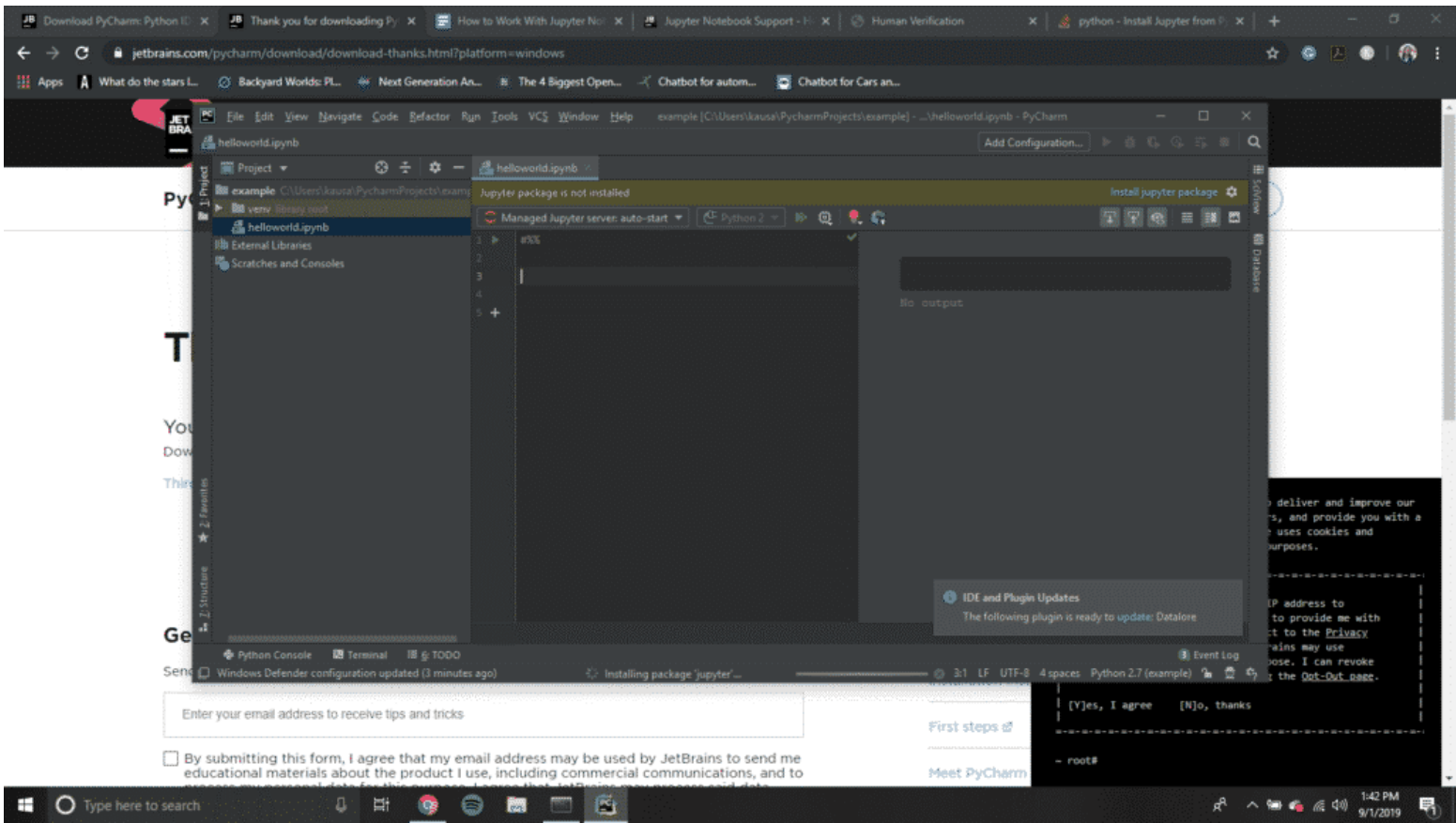
Приведите основные этапы работы с Jupyter notebook в IDE PyCharm и Visual Studio Code.

PyCharm:

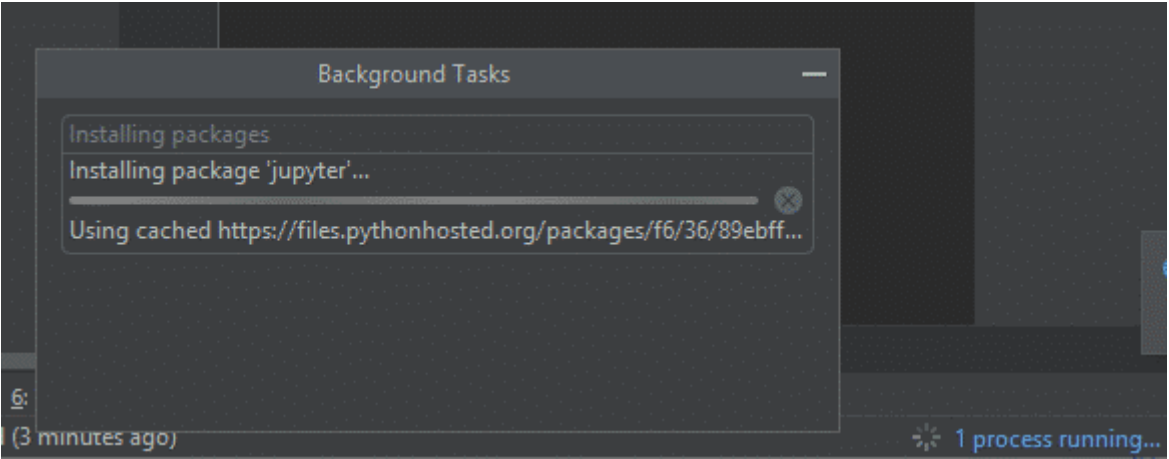
В настоящее время работа с Jupyter Notebook возможна только с лицензионной версией PyCharm Professional. PyCharm Professional не является бесплатным. Однако вы можете получить бесплатную лицензию, если вы связаны с образовательным учреждением.

Шаги работы:

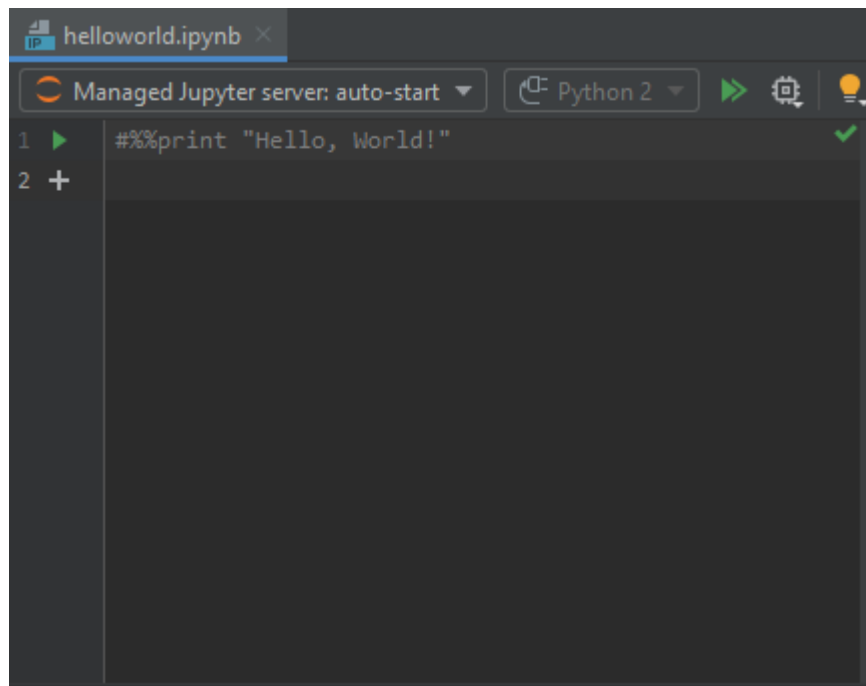
1. Сначала вы должны создать новый проект.
2. В этом проекте создайте новый файл `ipyunb`, выбрав `File> New...> Jupyter Notebook`. Это должно открыть новый файл записной книжки.
3. Если у вас не установлен пакет Jupyter Notebook, над вновь открытым файлом `ipyunb` появится сообщение об ошибке. Сообщение об ошибке гласит: «Пакет Jupyter не установлен», и у вас будет опция «Установить пакет jupyter» рядом с ним.



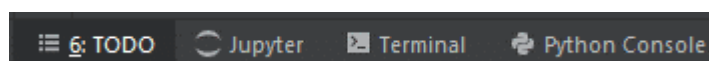
4. Нажмите «Установить пакет jupyter». Это запустит процесс установки, который вы можете просмотреть, щелкнув запущенные процессы в правом нижнем углу окна PyCharm.



5. Чтобы начать изучение Jupyter Notebook в PyCharm, создайте ячейки кода и выполните их.



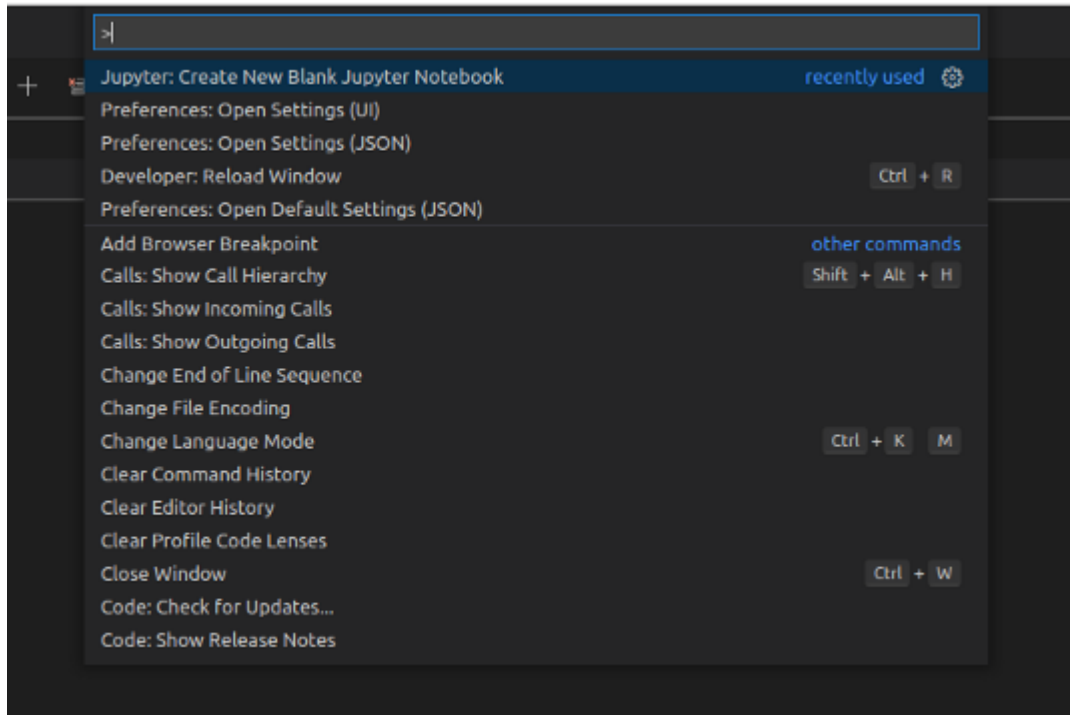
6. Выполните ячейку кода, чтобы запустить сервер Jupyter. По умолчанию сервер Jupyter использует порт 8888 на локальном хосте. Эти конфигурации доступны в окне инструментов сервера. После запуска вы можете просмотреть сервер над окном исходного кода, а рядом с ним вы можете просмотреть ядро, созданное как «Python 2» или «Python 3».
7. Теперь вы можете получить доступ к вкладке переменных в PyCharm, чтобы увидеть, как значения ваших переменных меняются при выполнении ячеек кода. Это помогает при отладке. Вы также можете установить точки останова в строках кода, а затем щелкнуть значок «Выполнить» и выбрать «Debug Cell» (или использовать сочетание клавиш Alt+Shift+Enter), чтобы начать отладку.
8. Следующие вкладки в нижней части окна PyCharm необходимы для использования Jupyter Notebook:



Visual Studio Code:

Удобная палитра команд VS Code позволяет выполнять все действия прямо с клавиатуры.

Чтобы создать новый Jupyter Notebook, запустите палитру команд (Ctrl+Shift+P) и выполните поиск new notebook. Первым результатом должен быть Jupyter: Create New Blank Jupyter Notebook. Вы также можете создать его, нажав на новый файл .ipynb, но горячие клавиши значительно повышают эффективность работы.



Обратите внимание, что блокноты, созданные VS Code, по умолчанию являются доверенными (trusted). С другой стороны, любые импортированные помечаются как not trusted, чтобы защитить пользователя от выполнения вредоносного кода. Таким образом, ставить пометку trust нужно вручную по запросу редактора перед выполнением.

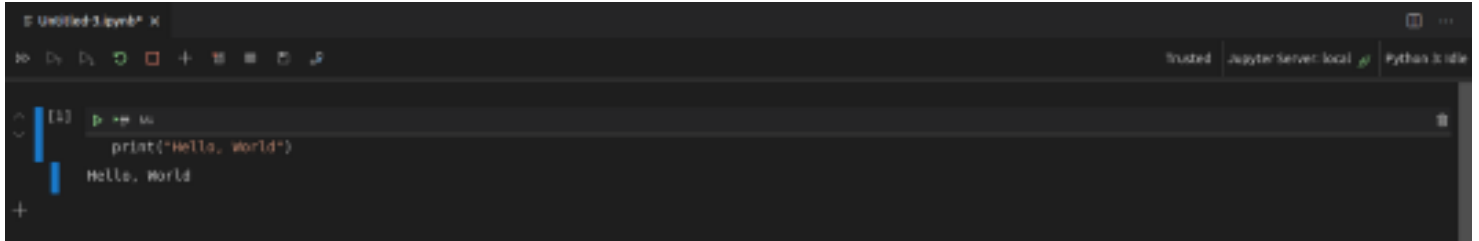
После создания блокнота нажмите на иконку save на верхней части панели инструментов, чтобы сохранить его в рабочем пространстве.



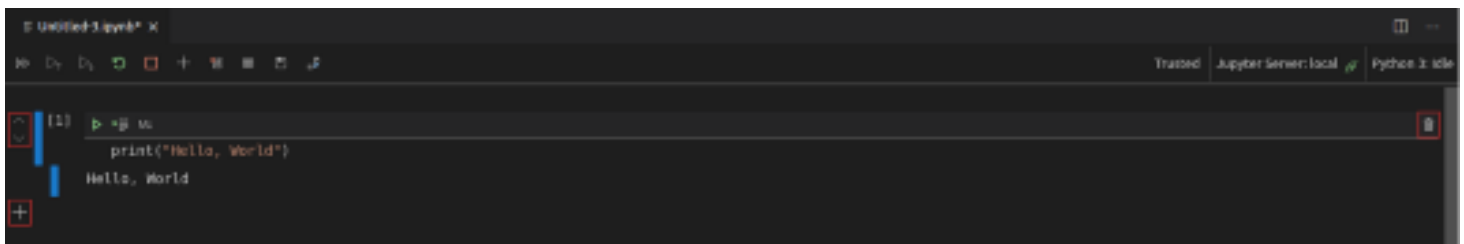
Теперь можно экспортировать созданный блокнот как скрипт Python или файл HTML/PDF, используя соответствующую иконку.



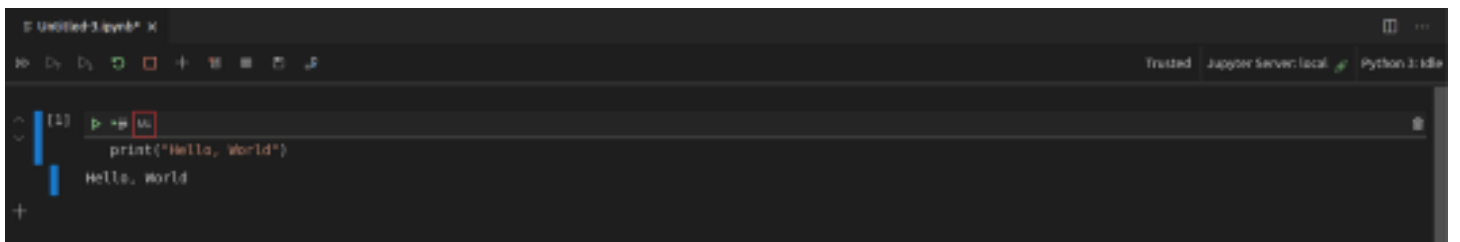
По умолчанию в новом блокноте появится пустая ячейка. Добавьте в нее код и выполните его с помощью *ctrl+enter*. Эта команда запустит выделенную ячейку. *shift+enter* выполняет то же действие, но при этом создает и выделяет новую ячейку ниже, а *alt+enter* выполняет выделенную, создает еще одну ниже, но при этом сохраняет метку на предыдущей.



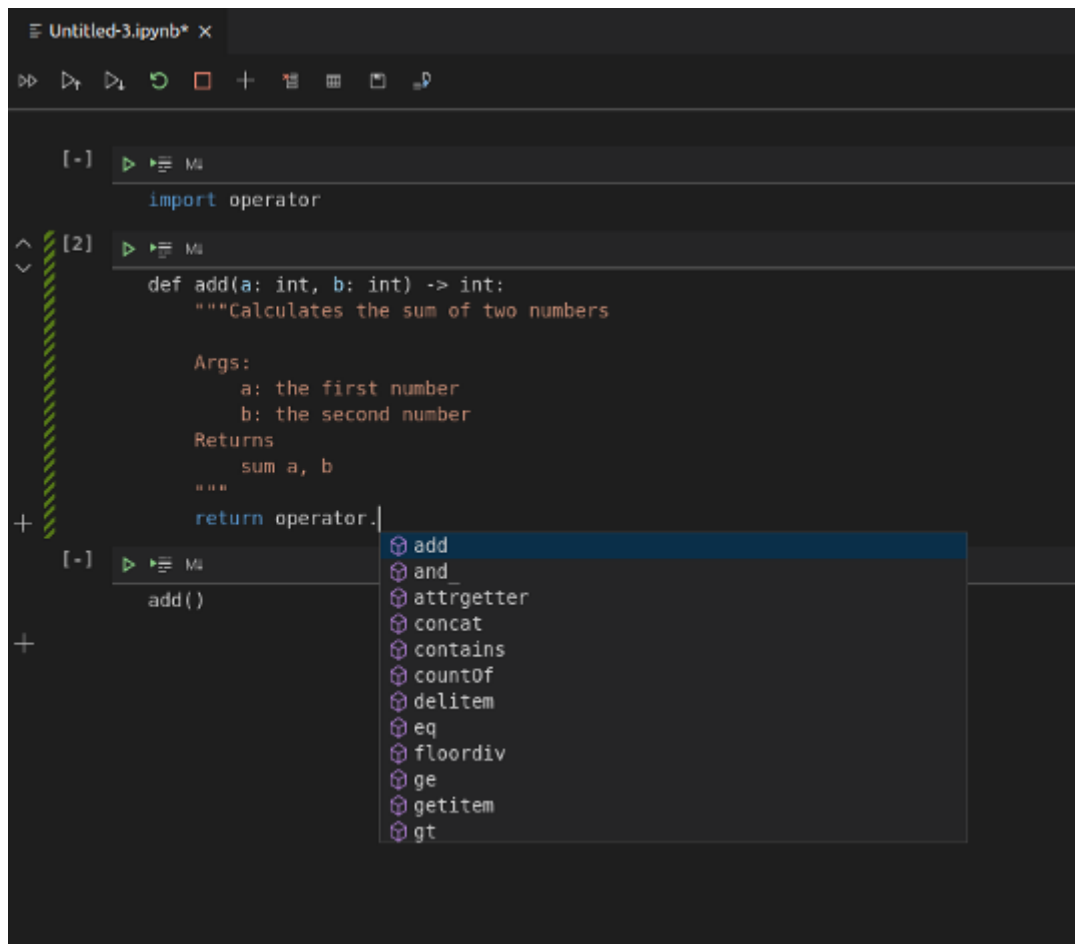
Иконка + добавляет новую ячейку для кода, а bin удаляет ее. Чтобы перемещать фрагменты вверх и вниз, воспользуйтесь соответствующими стрелками.



Изменить тип ячейки на markdown довольно просто: просто нажмите на иконку M, расположенную над кодом. Чтобы снова установить значение code, выберите значок {}. Выполнить эти действия также можно с помощью клавиш M и Y.



Встроенное автодополнение Intelligence — очень полезная функция VS Code. Редактор способен отображать списки членов, документацию методов и подсказки параметров.



The screenshot shows a Jupyter Notebook window titled 'Untitled-3.ipynb'. The code cell contains the following Python code:

```
[1] ▶ import operator

[2] ▶ def add(a: int, b: int) -> int:
    """Calculates the sum of two numbers

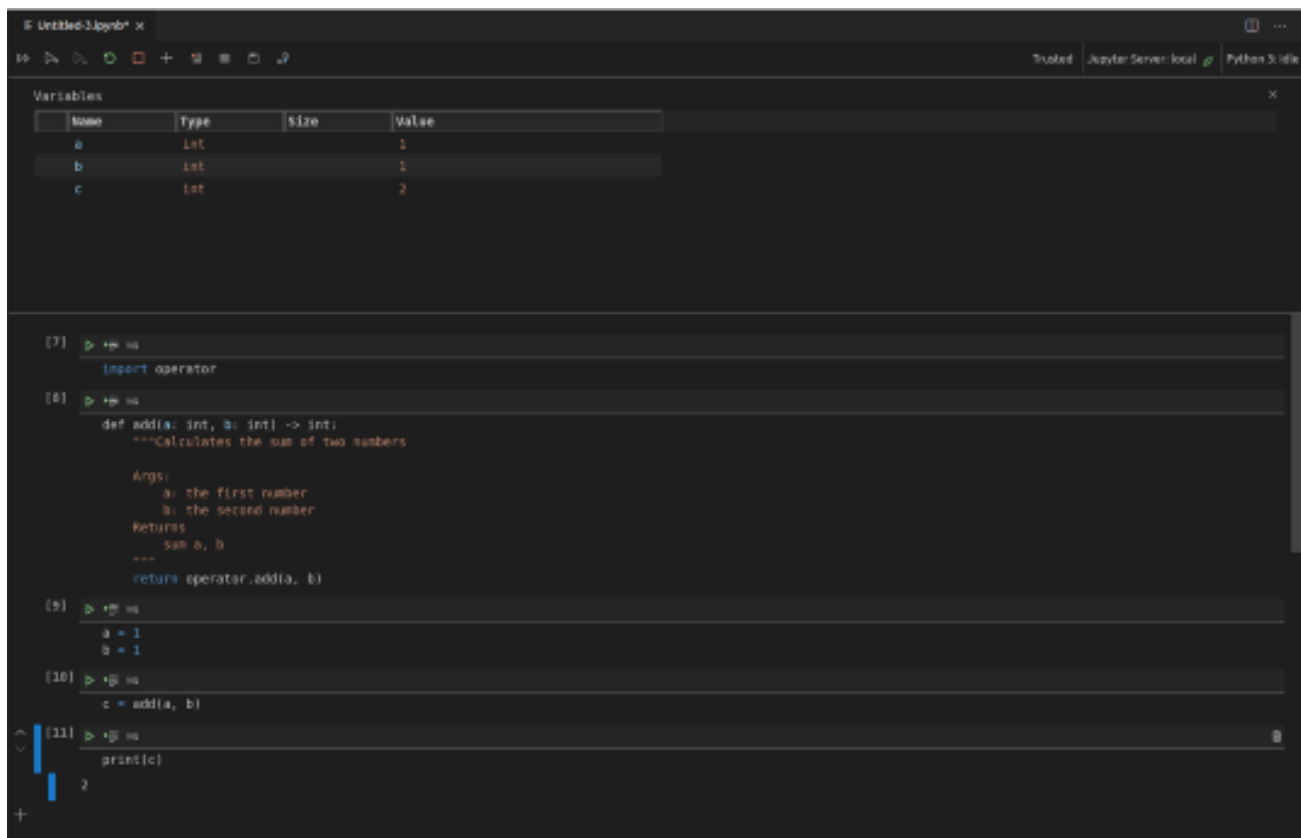
    Args:
        a: the first number
        b: the second number
    Returns:
        sum a, b
    """
    return operator.
```

A dropdown menu is open from the end of the last line of code, displaying a list of attributes from the `operator` module:

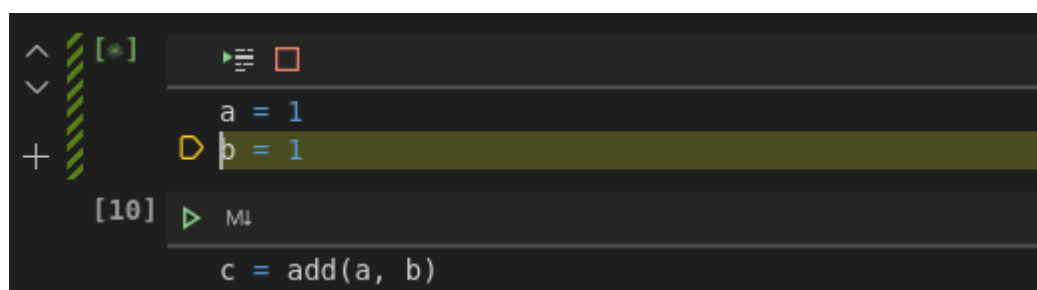
- add
- and_
- attrgetter
- concat
- contains
- countOf
- delitem
- eq
- floordiv
- ge
- getitem
- gt

Отслеживать находящиеся в памяти переменные помогают Variable Explorer и Data Viewer. Они позволяют просматривать, проверять и фильтровать переменные в текущей сессии Jupyter. Это сокращает проблемы, связанные с внеочередным исполнением, которые часто встречаются в блокнотах. Функция не решает их полностью, но выдает предупреждение.

После запуска кода нажмите на иконку Variables на верхней части панели инструментов. Вы увидите список текущих переменных, который будет обновляться автоматически при их использовании в коде.



Теперь рассмотрим опции отладки. Во-первых, расширение Jupyter для VS Code поддерживает построчное выполнение кода в ячейке. Просто нажмите на кнопку, расположенную рядом с иконкой play.



Вторая возможность для отладки дает весомую причину использовать Jupyter в VS Code. Вы можете просто экспортировать блокнот как скрипт Python и работать с ним прямо в отладчике VS Code, не переходя в другую среду.