

Отчёт по лабораторной работе №1 по дисциплине Компьютерный практикум по статистическому анализу данных

Julia. Установка и настройка. Основные принципы.

Шаповалова Диана Дмитриевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Подготовка инструментария к работе	6
2.2	Основы работы в блокноте Jupyter	9
2.3	Основы синтаксиса Julia на примерах	10
2.4	Задания для самостоятельной работы	11
3	Выводы	16
4	Список литературы	17

Список иллюстраций

2.1	Установка Chocolatey	7
2.2	Far уже был установлен	7
2.3	Установка полезных пакетов	8
2.4	Установка Anaconda	8
2.5	Запускаем Julia	9
2.6	Открываем Jupyter Lab	10
2.7	Примеры основ синтаксиса Julia	11
2.8	Примеры read, readline, readlines	12
2.9	Примеры print, println	12
2.10	Примеры show, write	12
2.11	Пример parse	13
2.12	Примеры базовых математических операций	14
2.13	Примеры операций над матрицами и векторами	15

Список таблиц

1 Цель работы

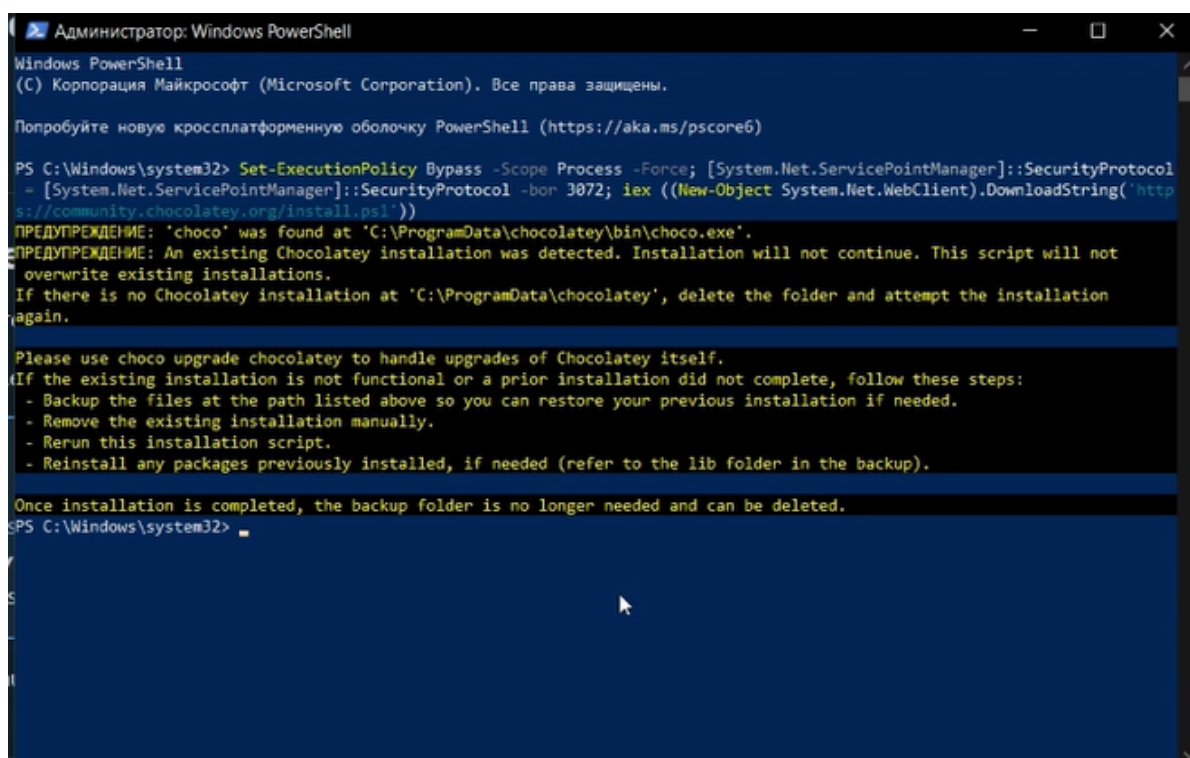
Основная цель работы — подготовить рабочее пространство и инструментарий для работы с языком программирования Julia, на простейших примерах познакомиться с основами синтаксиса Julia.

2 Выполнение лабораторной работы

2.1 Подготовка инструментария к работе

Установите Julia (<https://julialang.org/>) и Jupyter (<https://jupyter.org/>) под вашу операционную систему.

Для ОС типа Windows рекомендуется для установки использовать менеджер пакетов Chocolatey (<https://chocolatey.org/>), устанавливаемый через Administrative Shell. Далее рекомендуется посредством данного менеджера установить Far Manager, Notepad++, Julia, Anaconda Distribution (Python 3.x). (рис. 2.1 - рис. 2.4)



```
Администратор: Windows PowerShell
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/powershell)

PS C:\Windows\system32> Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol
- [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('http
s://community.chocolatey.org/install.ps1'))
ПРЕДУПРЕЖДЕНИЕ: 'choco' was found at 'C:\ProgramData\chocolatey\bin\choco.exe'.
ПРЕДУПРЕЖДЕНИЕ: An existing Chocolatey installation was detected. Installation will not continue. This script will not
overwrite existing installations.
If there is no Chocolatey installation at 'C:\ProgramData\chocolatey', delete the folder and attempt the installation
again.

Please use choco upgrade chocolatey to handle upgrades of Chocolatey itself.
If the existing installation is not functional or a prior installation did not complete, follow these steps:
- Backup the files at the path listed above so you can restore your previous installation if needed.
- Remove the existing installation manually.
- Rerun this installation script.
- Reinstall any packages previously installed, if needed (refer to the lib folder in the backup).

Once installation is completed, the backup folder is no longer needed and can be deleted.
PS C:\Windows\system32>
```

Рис. 2.1: Установка Chocolatey

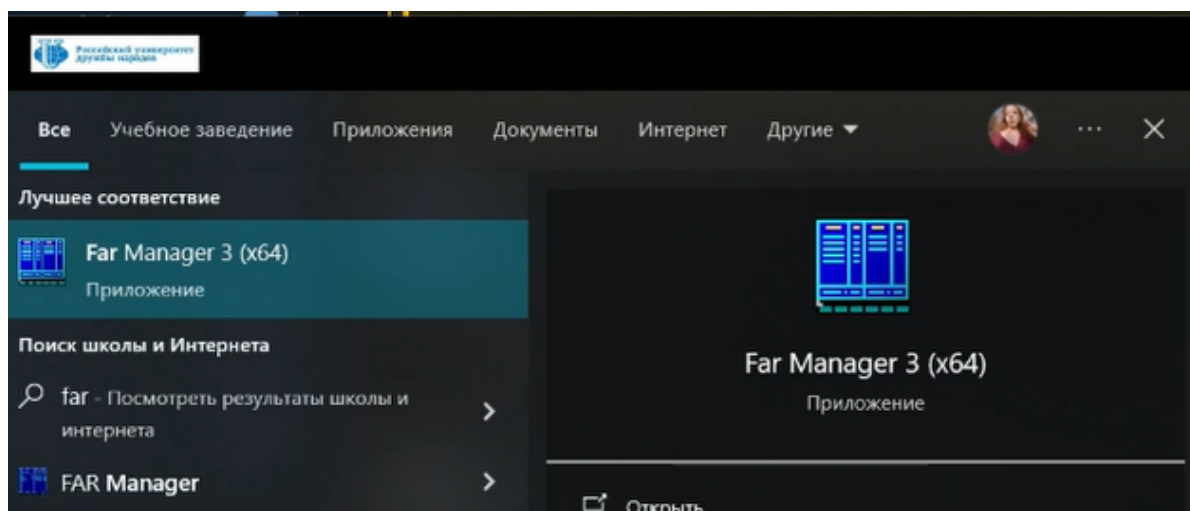


Рис. 2.2: Far уже был установлен

```

See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\Windows\system32> choco install julia
Chocolatey v2.3.0
Installing the following packages:
julia
By installing, you accept licenses for the packages.
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading chocolatey-compatibility.extension 1.0.0... 100%

chocolatey-compatibility.extension v1.0.0 [Approved]
chocolatey-compatibility.extension package files install completed. Performing other installation steps.
Installed/updated chocolatey-compatibility extensions.
The install of chocolatey-compatibility.extension was successful.
Deployed to 'C:\ProgramData\chocolatey\extensions\chocolatey-compatibility'
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading chocolatey-core.extension 1.4.0... 100%

chocolatey-core.extension v1.4.0 [Approved]
chocolatey-core.extension package files install completed. Performing other installation steps.
Installed/updated chocolatey-core extensions.
The install of chocolatey-core.extension was successful.
Deployed to 'C:\ProgramData\chocolatey\extensions\chocolatey-core'
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading Julia 1.10.5... 10%

```

Рис. 2.3: Установка полезных пакетов

```

See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\Windows\system32> choco install anaconda3
Chocolatey v2.3.0
Installing the following packages:
anaconda3
By installing, you accept licenses for the packages.
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading anaconda3 2024.10.0... 100%

anaconda3 v2024.10.0 [Approved]
anaconda3 package files install completed. Performing other installation steps.
The package anaconda3 wants to run 'chocolateyinstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): Y

WARNING: The Anaconda3 installation can take a long time (up to 30 minutes).
WARNING: Please be patient and let it finish.
WARNING: If you want to verify the install is running, you can watch the installer process in Task Manager
Downloading anaconda3 64 bit
from 'https://repo.anaconda.com/archive/Anaconda3-2024.10-1-Windows-x86_64.exe'
Progress: 0% - Saving 8.3 MB of 950.52 MB

```

Рис. 2.4: Установка Anaconda

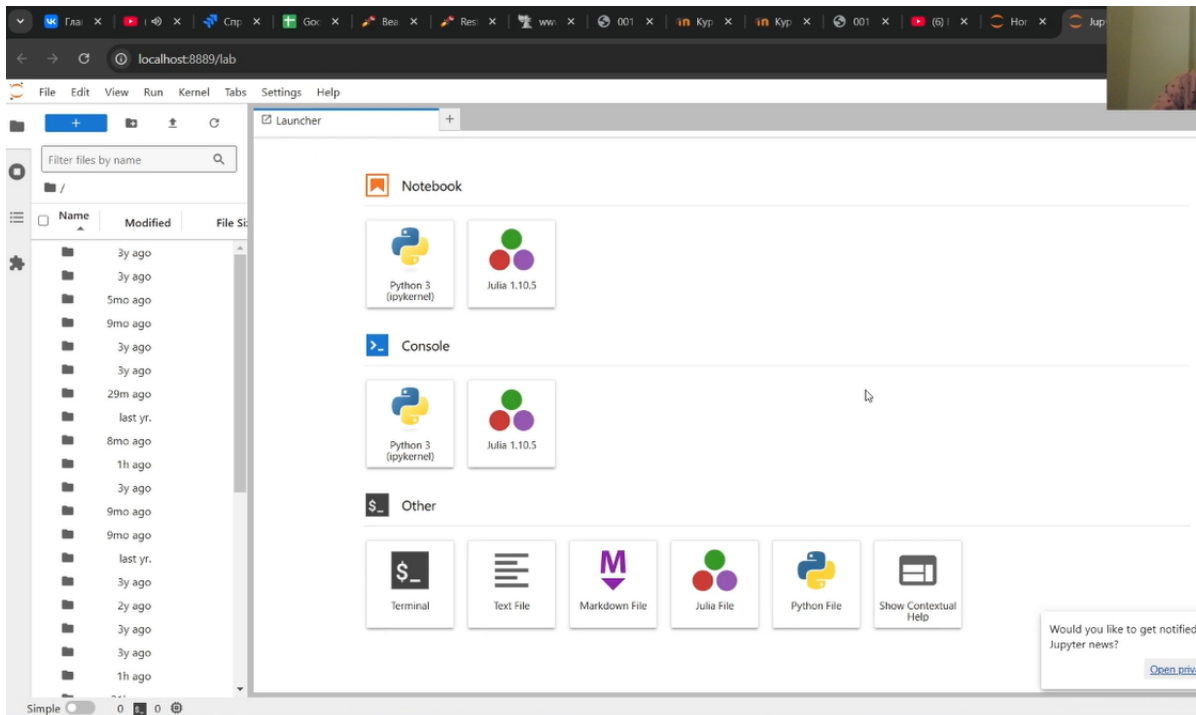


Рис. 2.6: Открываем Jupyter Lab

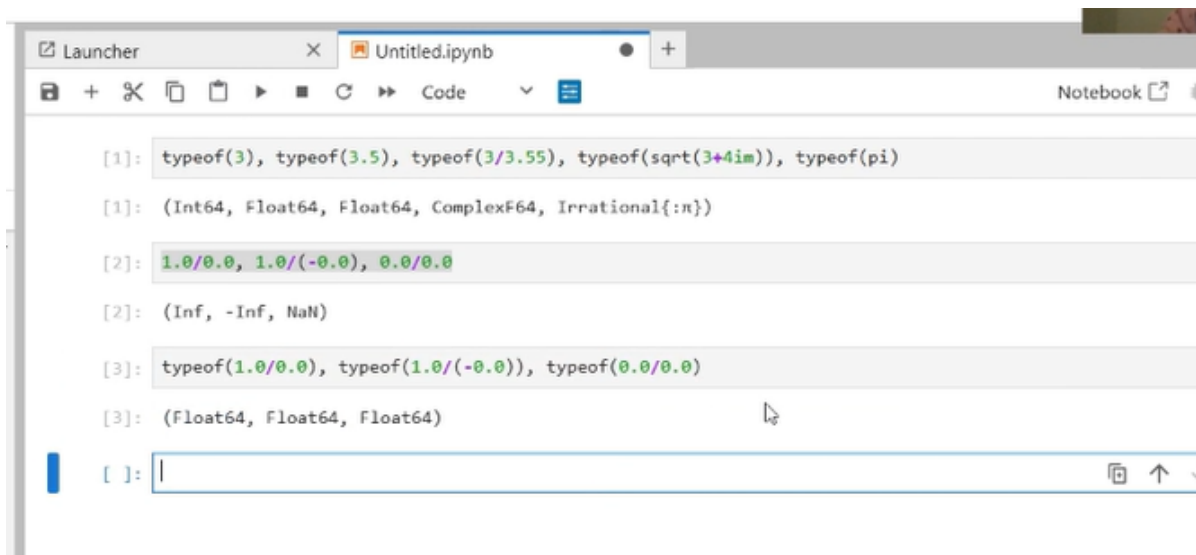
2.3 Основы синтаксиса Julia на примерах

В Julia преобразование типов можно реализовать или прямым указанием, например вещественное число 2.0 преобразовать в целое, а число 2 в символ: `Int64(2.0)`, `Char(2)`

или использовать обобщённый оператор преобразования типов `convert()`, например: `convert{Int64}(2.0)`, `convert{Char}(2)`

Преобразование 1 в булево `true`, 0 — в булево `false`: `Bool(1)`, `Bool(0)`

Для приведения нескольких аргументов к одному типу, если это возможно, используется оператор `promote()`, например: `promote{Int8, Float16}(1, 4.5)`, `promote{Float32, Int8}(4.1, 1)`



```
[1]: typeof(3), typeof(3.5), typeof(3/3.55), typeof(sqrt(3+4im)), typeof(pi)
[1]: (Int64, Float64, Float64, ComplexF64, Irrational{::N})

[2]: 1.0/0.0, 1.0/(-0.0), 0.0/0.0
[2]: (Inf, -Inf, NaN)

[3]: typeof(1.0/0.0), typeof(1.0/(-0.0)), typeof(0.0/0.0)
[3]: (Float64, Float64, Float64)
```

Рис. 2.7: Примеры основ синтаксиса Julia

2.4 Задания для самостоятельной работы

1. Изучите документацию по основным функциям Julia для чтения / записи / вывода информации на экран: `read()`, `readline()`, `readlines()`, `readdlm()`, `print()`, `println()`, `show()`, `write()`. Приведите свои примеры их использования, поясняя особенности их применения. (рис. 2.8 - рис. 2.10)

Функция `read()` используется для чтения содержимого файла или потока. Она позволяет прочитать данные в бинарном или текстовом формате.

Функция `readline()` читает одну строку из файла или стандартного потока ввода.

Функция `readlines()` считывает все строки файла и возвращает их в виде массива строк.

Функция `readdlm()` (`read delimited`) используется для чтения данных из файла с разделителями (например, CSV) в виде таблицы.

```
[4]: file = open("hello.txt", "r")
content = read(file, String)
println(content)

Hello, world! D

[8]: file = open("hello.txt", "r")
first_line = readline(file)
println(first_line)

Hello, world! first line

[9]: file = open("hello.txt", "r")
all_lines = readlines(file)
println(all_lines)

["Hello, world! first line", "Hello, world! Second line"]
```

Рис. 2.8: Примеры read, readline, readlines

```
["Hello, world! first line", "Hello, world! Second line"]

[11]: print("это ")
      print("строка.")

это строка.

[12]: println("это ")
      println("2 строки.")

это
2 строки.
```

Рис. 2.9: Примеры print, println

```
2 строки.

[17]: arr = [1, 2, 3]
      show(arr)

[1, 2, 3]

[18]: write("hello.txt", "Зя строка")

[18]: 16

[31]: write("hello.txt", "3-я строка")

[31]: 17
```

Рис. 2.10: Примеры show, write

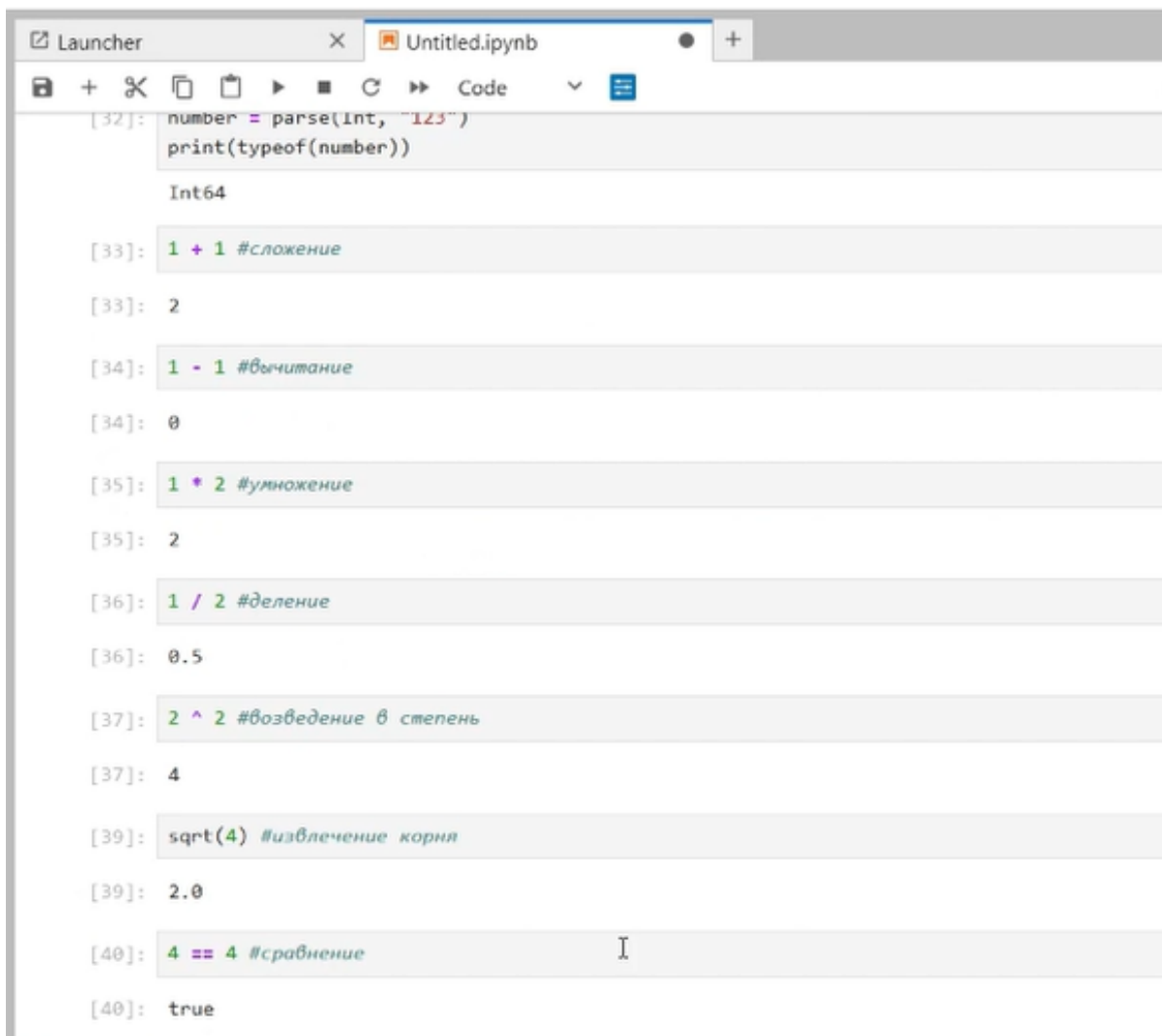
2. Изучите документацию по функции `parse()`. Приведите свои примеры её использования, поясняя особенности её применения. (рис. 2.11)

Функция `parse()` в Julia используется для преобразования строки в значение заданного типа. Она особенно полезна, когда нужно перевести текстовое представление числа или другого типа данных в соответствующий числовой или логический формат.

```
[32]: number = parse{Int, "123"}  
      print(typeof(number))  
  
      Int64
```

Рис. 2.11: Пример `parse`

3. Изучите синтаксис Julia для базовых математических операций с разным типом переменных: сложение, вычитание, умножение, деление, возведение в степень, извлечение корня, сравнение, логические операции. Приведите свои примеры с пояснениями по особенностям их применения. (рис. 2.12)



The screenshot shows a Jupyter Notebook window titled 'Untitled.ipynb'. The interface includes a toolbar with icons for saving, adding cells, deleting, copying, pasting, running, and code completion. The notebook contains several code cells, each with a prompt (e.g., [32]:, [33]:) and a corresponding output. The operations shown are: parsing a string to an integer, addition, subtraction, multiplication, division, exponentiation, square root calculation, and a comparison operation. Comments in Russian are provided for the arithmetic operations.

```
[32]: number = parse(Int, "123")
      print(typeof(number))
      Int64

[33]: 1 + 1 #сложение
[33]: 2

[34]: 1 - 1 #вычитание
[34]: 0

[35]: 1 * 2 #умножение
[35]: 2

[36]: 1 / 2 #деление
[36]: 0.5

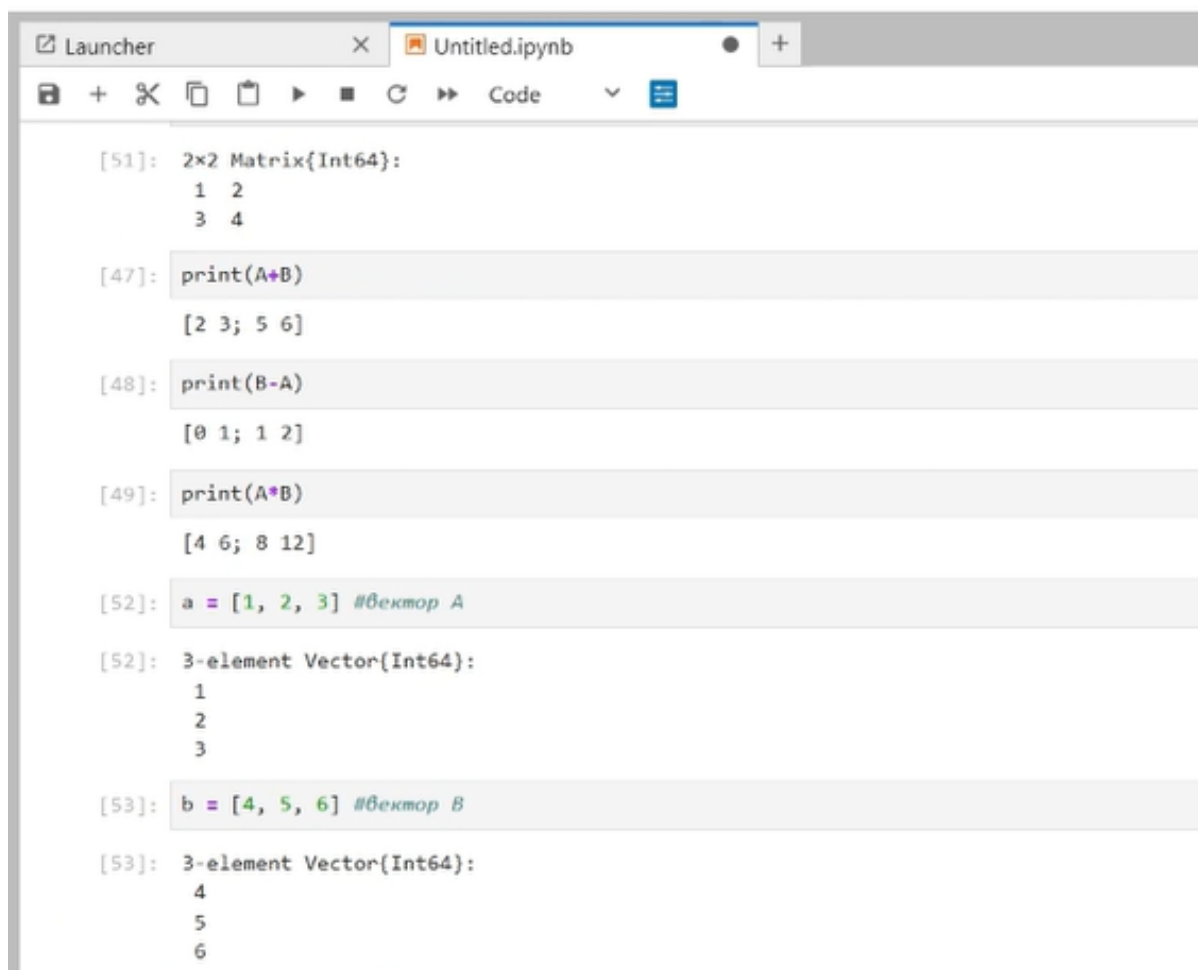
[37]: 2 ^ 2 #возведение в степень
[37]: 4

[39]: sqrt(4) #извлечение корня
[39]: 2.0

[40]: 4 == 4 #сравнение
[40]: true
```

Рис. 2.12: Примеры базовых математических операций

4. Приведите несколько своих примеров с пояснениями с операциями над матрицами и векторами: сложение, вычитание, скалярное произведение, транспонирование, умножение на скаляр. (рис. 2.13)



The screenshot shows a Jupyter Notebook window titled 'Untitled.ipynb'. The interface includes a toolbar with icons for file operations and a 'Code' dropdown menu. The notebook contains several code cells with the following content:

```
[51]: 2x2 Matrix{Int64}:  
      1 2  
      3 4  
  
[47]: print(A+B)  
      [2 3; 5 6]  
  
[48]: print(B-A)  
      [0 1; 1 2]  
  
[49]: print(A*B)  
      [4 6; 8 12]  
  
[52]: a = [1, 2, 3] #вектор A  
  
[52]: 3-element Vector{Int64}:  
      1  
      2  
      3  
  
[53]: b = [4, 5, 6] #вектор B  
  
[53]: 3-element Vector{Int64}:  
      4  
      5  
      6
```

Рис. 2.13: Примеры операций над матрицами и векторами

3 Выводы

Мы подготовили наше рабочее пространство и инструментарий для работы с языком программирования Julia, а также познакомились с основами синтаксиса Julia

4 Список литературы

[1] Julia Documentation: <https://docs.julialang.org/en/v1/>