

Издательство Прометей

IB Institute
Международный Банковский Институт имени Анатолия Собчака

Машинное обучение в финансах

Учебник для магистратуры под общей редакцией
С.Ю. Богатырева

#Data Science for Investment Professionals Certificate
#Fintech #Data Science
#Investment Management #Quantitative Methods

УДК 004.8-338.27
ББК 32.813-65.261
М38

Рецензенты:

Зададаев Сергей Алексеевич, кандидат физико-математических наук, профессор, руководитель Департамента математики Финансового университета при Правительстве Российской Федерации, Почетный работник высшего профессионального образования Российской Федерации;

Мунерман Илья Викторович, кандидат экономических наук, директор Исследовательского подразделения «Интерфакс-ЛАБ», Группа «Интерфакс».

Машинное обучение в финансах: Учебник для магистратуры / С.Ю.Богатырев, А.А. Помулев, А.В. Затевахина [и др.] ; под ред. С.Ю. Богатырева — М.: Прометей, 2024. — 224 с.

М38

ISBN 978-5-00172-572-5

Учебник предназначен для проведения занятий по дисциплине «Машинное обучение в финансах». В учебнике описано, как работать с основными инструментами машинного обучения: Python, Pandas. Приведены примеры кода для загрузки, обработки и анализа информации с финансовых рынков. Описаны современные направления использования машинного обучения в финансовой сфере. Очерчены перспективные направления разработки средств машинного обучения на финансовых рынках.

Материал учебника предоставляет читателям практические знания в области методов работы с данными и основ машинного обучения, а также того, как они используются в инвестиционном процессе. На примере виртуальных кейсов читатели получают практический опыт программирования, основанный на реалистичных сценариях, применяют концепции машинного обучения к реальным инвестиционным проблемам и учатся ясно объяснять их неспециалистам и клиентам.

Учебник предназначен для магистрантов, обучающихся по направлениям подготовки «Экономика», «Финансы и кредит», слушателей сокращенных программ, программ профессиональной переподготовки финансистов. Материалы учебника могут использоваться для подготовки к экзамену на получение сертификата CFA Data Science for Investment Professionals Certificate.

ISBN 978-5-00172-572-5

© Коллектив авторов, 2024
© Издательство «Прометей», 2024

Авторский коллектив

Богатырев Семен Юрьевич — Введение, Главы 1 и 3; параграфы 2.1, 2.3, Глава 4, Заключение (в соавторстве);

Помулев Александр Александрович — параграф 2.2;

Затевахина Анна Васильевна — Вопросы задания к главе 1, параграф 4.2 (в соавторстве);

Круглова Инна Александровна — Вопросы задания к главе 1, параграф 4.1 (в соавторстве);

Барабанова Марина Ивановна — Вопросы задания к главе 1, параграф 4.3 (в соавторстве);

Тегин Алексей Владленович — параграф 2.3 (в соавторстве);

Солодовников Максим Алексеевич — параграф 2.1 (в соавторстве);

Шашина Ирина Александровна — параграф 2.1 (в соавторстве);

Матросов Сергей Викторович — параграф 2.3, Заключение (в соавторстве).

Содержание

Введение. 6

Глава 1. **ИНСТРУМЕНТЫ МАШИННОГО ОБУЧЕНИЯ** 8

1.1. Установка среды программирования 8
1.2. Язык программирования Питон. Основы Питона . . . 11
1.3. Работа с библиотеками в Пандас. Pandas 36
Вопросы и задания к главе 1 104

Глава 2. **ИНСТРУМЕНТЫ МАШИННОГО ОБУЧЕНИЯ: ПРИМЕНЕНИЕ В ФИНАНСАХ** 105

2.1. Питон для обработки финансовой информации 105
2.2. Работа с финансовой информацией в Питоне: продвинутый уровень 112
2.3. Приложения NLP в управлении активами. 148
Вопросы и задания к главе 2 161

Глава 3. **МАШИННОЕ ОБУЧЕНИЕ И ПРИМЕНЕНИЕ DATA SCIENCE В ИНВЕСТИЦИЯХ**. 162

3.1. Машинное обучение в работе современного финансиста. 162
3.2. Показатели, инструменты и технологии машинного обучения в финансово-инвестиционной деятельности . . 167
3.3. Практические приложения машинного обучения в инвестициях 185
Вопросы и задания к главе 3 199

Глава 4. **ЭТИЧЕСКИЕ ПРАВИЛА ПРИМЕНЕНИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ПРИ УПРАВЛЕНИИ ИНВЕСТИЦИЯМИ** 200

4.1. Общие понятия этики финансиста при применении искусственного интеллекта 200

4.2. Критерии добросовестного применения технологий машинного обучения и искусственного интеллекта в практике финансово-инвестиционного консультирования 202
4.3. Регуляторная база применения технологий искусственного интеллекта в финансово-экономической сфере 207
Вопросы и задания к главе 4 213

Заключение 214

Список литературы 216

Глава 1. ИНСТРУМЕНТЫ МАШИННОГО ОБУЧЕНИЯ

Code first, think after!
Сначала программируй, потом думай!

1.1. Установка среды программирования

Точка запуска инструментов искусственного интеллекта (ИИ) для финансовых исследований.

Среда для работы — Jupiter notebook.

Первый используемый инструмент — collab. Это облачный Jupiter, бесплатно предоставляемый Google.

Для запуска этого инструмента ИИ в строке браузера надо набрать: https://colab.research.google.com/?utm_source=scs-index

Результат работы команды — на рисунке 1:

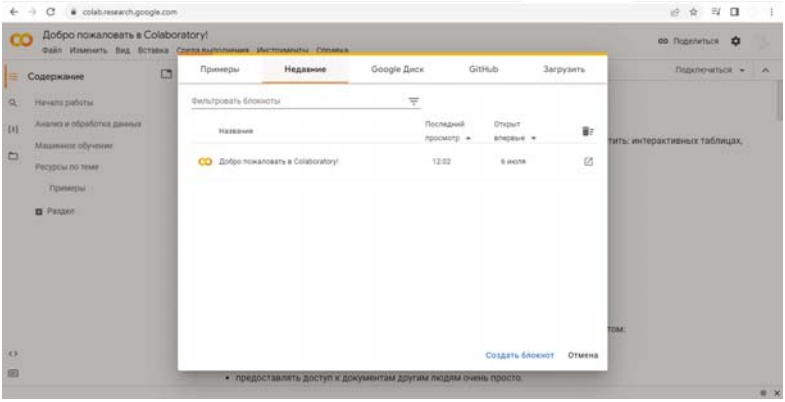


Рисунок 1. Начальное меню работы с Colab

Для удобства работы лучше сразу добавить его в закладки браузера.

В пункте меню «Недавние» система показывает недавно использовавшиеся ноутбуки. Чтобы создать новый, надо нажать кнопку «Создать блокнот».

После этого запускается окно для создания кода Питона — рисунок 2.



Рисунок 2. Окно создания нового ноутбука

Colab входит в линейку продуктов Google — гугл документы. Это значит, что так же, как и всеми другими продуктами, создаваемыми в гугл: текстовыми файлами, электронными таблицами, ноутбуком можно поделиться с другими пользователями, коллегами — рисунок 3.

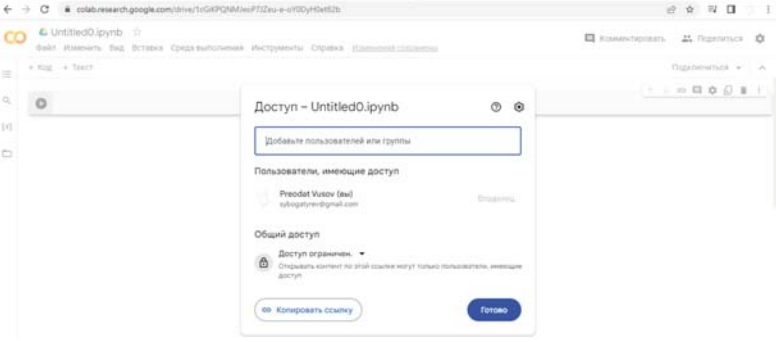


Рисунок 3. Расшаривание нового ноутбука

На этом этапе передачи содержащейся в ноутбуке информации, необходимо проверить, содержит ли она персональные данные, которые нельзя передавать в другую организацию, на другие серверы. Эта проблема решается локальной установкой Jupiter к себе на компьютер.

Для целей научных исследований и в учебной работе при обработке массивов информации с финансовых рынков с публичным доступом это не требуется, и использование Colab представляется наиболее эффективным с точки зрения инфраструктуры.

Colab лучше запускать на английском языке. Это связано с неточностями перевода, интегрированностью процесса программирования в Colab в устойчивую терминологию используемых терминов и команд. Чтобы сменить язык с русского на английский при первом запуске Colab надо зайти в пункт меню Справка и выбрать пункт меню Показать на английском языке — рисунок 4.

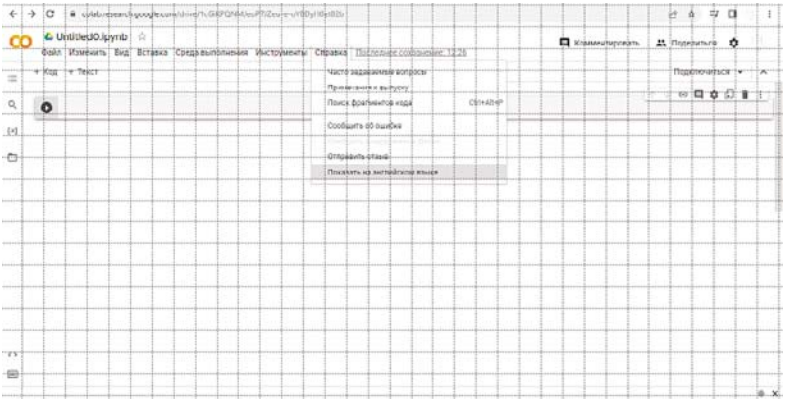


Рисунок 4. Смена языка в Colab

Следующее рекомендуемое изменение настроек по умолчанию — это смена фона с белого на черный, чтобы не уставали глаза — рисунок 5, пункт меню Tools/Settings.

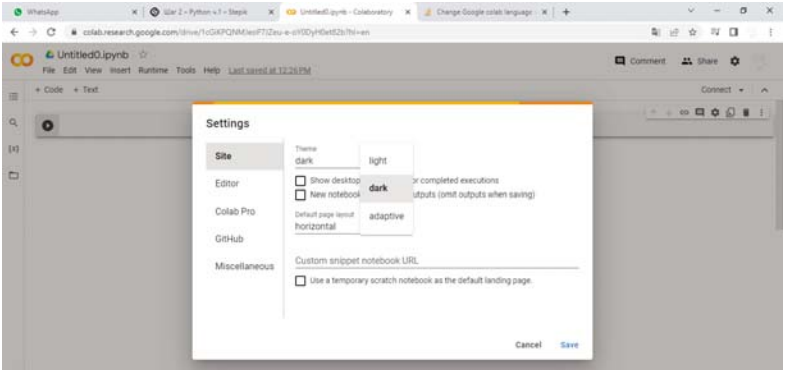


Рисунок 5. Смена фона

В браузере Google Chrome для этих же целей можно установить надстройку Dark Reader, которая будет отображать открываемые сайты на темном фоне, сберегая глаза.

1.2. Язык программирования Питон. Основы Питона

При описании основ языка программирования Питон необходимо сделать важное замечание: аналитик и инженер по данным data scientist (дата-сиенсист) — не программисты. Использование Питона определяется не желанием научиться на нём программировать, но тем фактом, что на этом достаточно почтенном по возрасту (язык был создан в 1969 году) языке программирования создано огромное количество библиотек, уже готовых программ, с помощью которых можно выполнять свои задачи.

Таким образом, для аналитика Питон представляет собой интерфейс, где вместо кнопок и пунктов меню — команды и правила их написания в определённых языком последовательностях. Выбирая для нужных целей кусочек уже созданного кода, меняя определённые параметры в нём, аналитик решает свои собственные задачи. При таком взгляде на Питон он представляет собой платформу, содер-

жащую набор готовых программ, и вся задача аналитика — научиться их правильно и эффективно использовать.

Если в командную строку Питона начать вводить операции, они сразу выполняются в программе — рисунок 6.



Рисунок 6. Выполнение простейших команд в Colab

Конечно, этим работа программы не ограничивается. Для усложнения операций вводятся переменные. Переменные — символьные обозначения для удобства создания формул. При этом формула — универсальное расчётное представление закреплённых в формуле закономерностей. В каждую из переменных подставляется значение для осуществления в дальнейшем определённых в формуле действий.

При программировании на Питоне переменные лучше не называть односложными буквами и буквенными сочетаниями, несущими определённую смысловую нагрузку.

Пример создания переменной и простейшей операции (умножение) с созданной переменной — на рисунке 7.

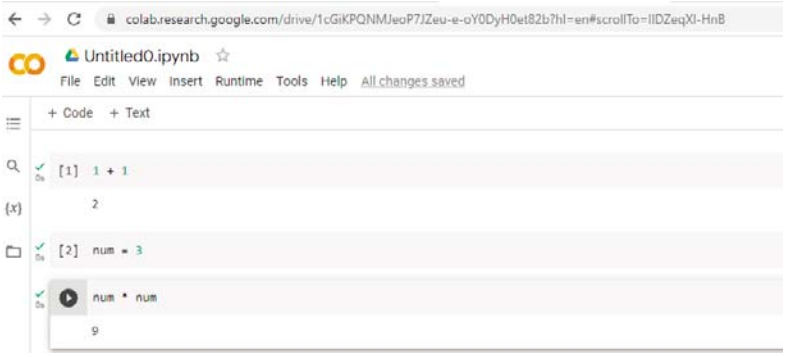


Рисунок 7. Создание переменной и умножение созданных переменных

При изменении значения переменной оператор, использующий её значение, пересчитается — рисунок 8.

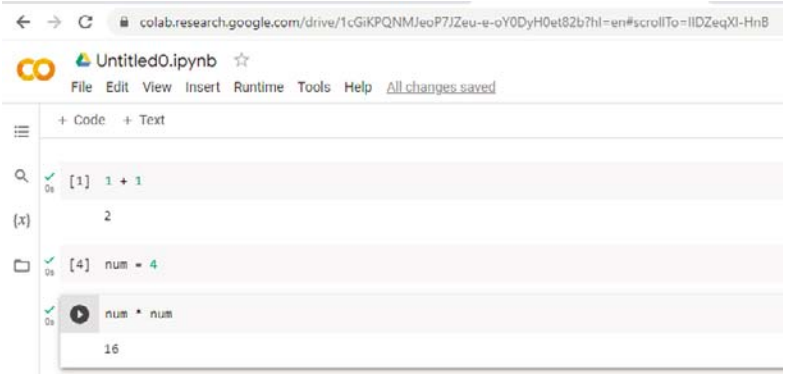


Рисунок 8. Пересчёт оператора при изменении переменной

Чем содержательнее названы переменные, тем легче затем понимать код, созданный с их использованием.

Помимо чисел Питон работает с другими типами данных, а именно: строка — сочетания буквенных символов. Задаваемую переменную лучше помещать в двойные кавычки, тогда внутри строки можно использовать одинарные кавычки, как показано на рисунке 9.



Рисунок 9. Определение строки

Создание строк при помощи трёх точек — расширение возможностей по представлению данных (рисунок 10).

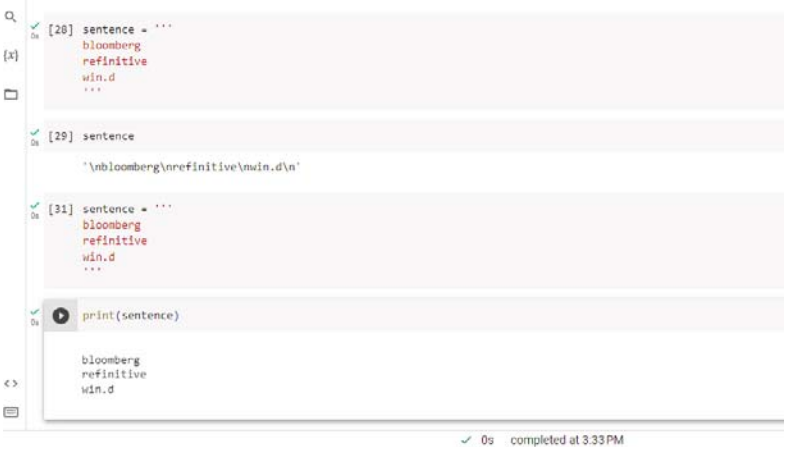


Рисунок 10. Создание строк при помощи трёх точек

Любое число можно перевести в строку — рисунок 11:



Рисунок 11. Превращение числа в строку

Использовав эту команду можно создавать строки с добавлением переменных — рисунок 12:

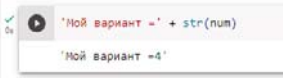


Рисунок 12. Создание строки с добавлением переменных

Другой способ сделать то же самое — применить оператор f (рисунок 13):

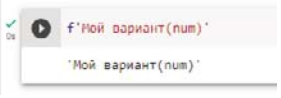


Рисунок 13. Применение оператора f

При изменении значения переменной num изменится значение строки.

При работе со строкой можно задать получение любого её символа, нумерация начинается с нуля — рисунок 14:



Рисунок 14. Возвращение первой переменной из строки Bloomberg

Или в начале работы с ноутбуком, или когда уже становится понятно, какие цели он решает, следует переименовать его, как показано на рисунке 15:

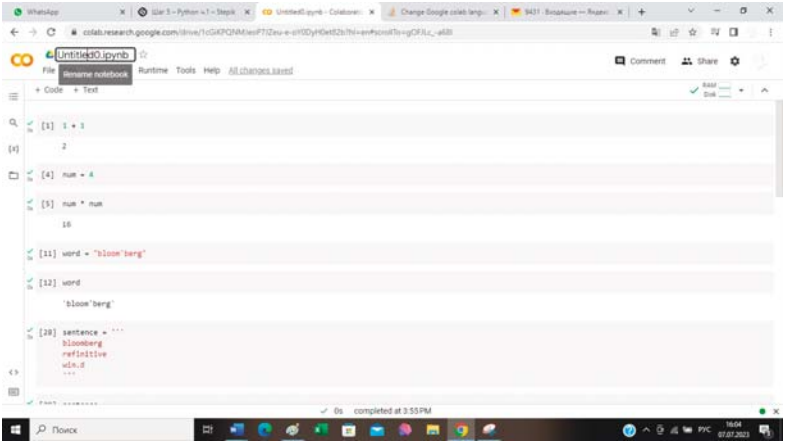


Рисунок 15. Переименование ноутбука

Для аккуратного стильного наименования файлов не следует использовать знак минус и пробелы, символы помимо латиницы. Тоже самое рекомендуется делать с переменными и со строками: вместо знака минус следует использовать нижнее подчеркивание. В названии переменной вместо пробела следующее слово можно начать с большой буквы (CamelCase). Код в Питоне приобретает профессио-

нальный вид, когда при обозначении переменных используется snake case — слова разделяются нижним подчеркиванием.

Тип данных «список».

При указании курсором на нужный оператор зайти в пункт меню Текстовая ячейка и выбрать Создание текстовой ячейки. Набрать — Списки. Рисунок 16.

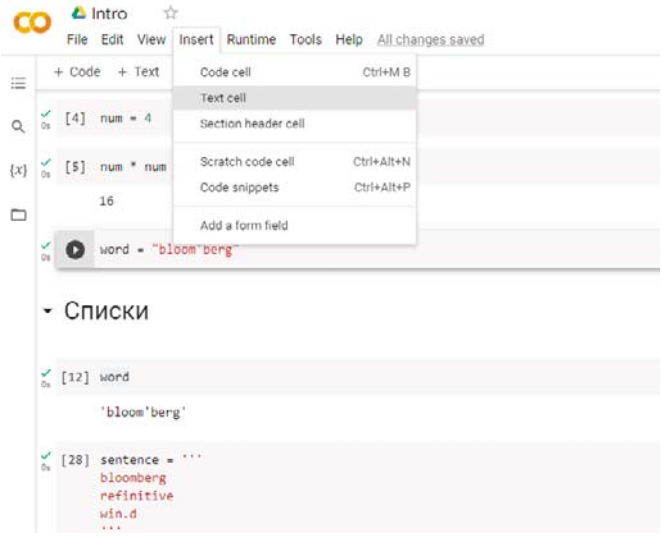


Рисунок 16. Создание списка

Зайдя в оператор переменных выбрать Создание текстовой ячейки и создать раздел «Переменные» — рисунок 17.

При этом справа в меню отобразятся созданные разделы оглавления.

Список содержит в себе ряд переменных. Создание списка — рисунок 18.

В списке хранятся несколько чисел, и они могут быть вызваны обращением к их порядковому номеру, также как в случае со строкой — рисунок 19.

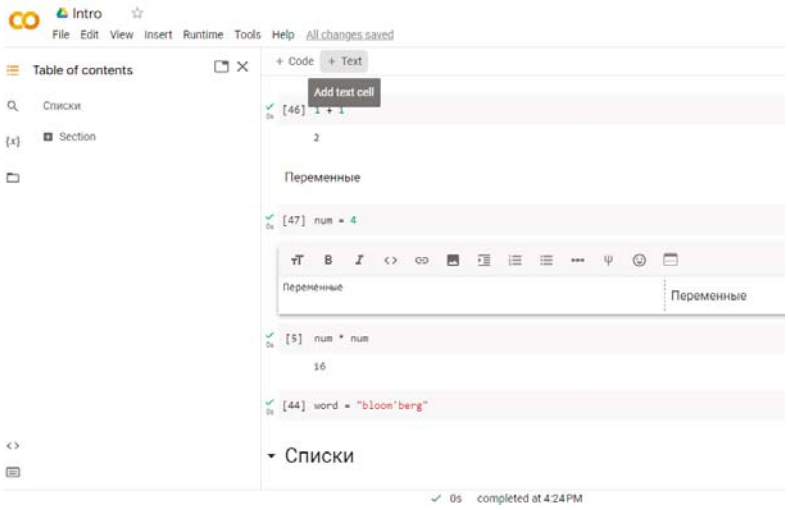


Рисунок 17. Создание оглавления

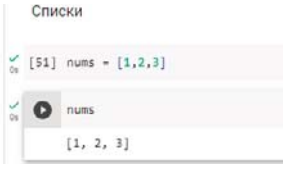


Рисунок 18.



Рисунок 19. Работа со списком

В списках хранятся однотипные переменные. Работа с каждым элементом списка уже осложняется, так как их много. Для того чтобы не работать с каждым элементом списка отдельно, при работе со списками применяются циклы. Цикл в Питоне позволяет делать повторяющиеся действия, например умножить все элементы списка на 3 — рисунок 20.



Рисунок 20. Умножение всех элементов списка на 3 с помощью цикла в Питоне

Оператор range в данном случае указывает на длину обрабатываемого списка. При использовании в отдельности он позволяет генерировать список в соответствии с указаниями команды.

Работа команды range и представление её в виде списка — на рисунке 21.



Рисунок 21. Работа с range

Таким образом в Питоне генерируются последовательные числа. Генерацию чисел при помощи этой команды можно проводить с разным шагом, как указано на рисунке 22: генерация чисел от 0 до 100 с шагом 5.



Рисунок 22. Пошаговая генерация чисел

Генерация чисел от 10 до 0 на рисунке 23.

Чтобы распечатать каждый элемент списка, команду на рисунке 20 можно преобразовать следующим образом — рисунок 24.

```
list (range(10,-1,-1))
```
```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

Рисунок 23. Обратная генерация чисел

```
for i in range(len(nums)):
 print(nums[i])
```
```
1
2
3
```

Рисунок 24. Распечатка каждого элемента списка

Другой способ перебора списка — рисунок 25.

Главное в работе этой команды — ставить в скобки то же самое слово, которое указано после оператора for, причем оно может быть любым. Внутри скобок можно добавить любой оператор действий с элементами списка. На рисунке 26 указано умножение на 10.

```
for num in nums:
 print (num)
```
```
1
2
3
```

Рисунок 25. Перебор списка

```
for num in nums:
 print (num * 10)
```
```
10
20
30
```

Рисунок 26. Действия с элементами списка при переборе

Важный и полезный при работе с большими списками финансово-экономических показателей оператор позволяет при переборе данных списка выводить их вместе с их порядковыми номерами — рисунок 27.

```
for i in range(len(nums)):
 print(i,nums[i])
```
```
0 1
1 2
2 3
```

Рисунок 27. Вывод данных списка с порядковыми номерами

Вторая форма записи этого же действия с помощью команды enumerate — рисунок 28.

```
for i, num in enumerate(nums):
  print(i,num)
```
```
0 1
1 2
2 3
```

Рисунок 28. Вывод данных списка с порядковыми номерами

В Питоне данные можно представлять не только в виде списка, но и в виде кортежа. При создании кортежа данные заключают в круглые скобки. С точки зрения логики опера-

ций, не существует различий между ними над этими двумя формами представления данных. Главное отличие кортежа от списка состоит в том, что из кортежа добавление элементов списка производится командой append, как показано на рисунке 29.

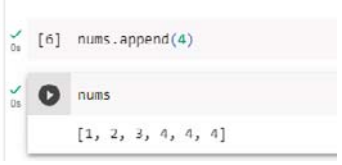


Рисунок 29. Добавление элемента списка

На рисунке цифра 4 была добавлена четыре раза последовательным обновлением команды append. Для обновления и запуска команды надо нажать на стрелочку в кружочке.

Удаление из списка последнего элемента осуществляется командой pop — рисунок 30.

С кортежем эта команда работать не будет.

Создание кортежа — команда nums_tuple = (1, 2, 3) и попытка добавления в него нового элемента 4 — рисунок 31.

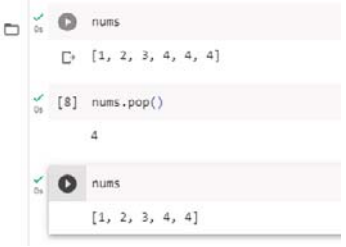


Рисунок 30. Удаление элемента списка



Рисунок 31. Создание кортежа и попытка добавления в него элемента

В кортеже хранят данные, имеющие определённый формат, например координаты. Тогда это будет особенно удобно, так как если координат может быть только две, то их нельзя будет изменить.

Слайсинг (нарезка).

При помощи слайсинга из списка можно «вырезать» те или иные элементы.

На рисунке 32 показано, как из созданного списка: берётся первый элемент списка, отбор первых двух элементов (при этом ноль может не прописываться в коде), отбор последнего элемента списка, отбор всех элементов кроме первого:



Рисунок 32. Слайсинг списка – часть 1

На рисунке 33 изображены следующие операции слайсинга — отбор всех элементов кроме последнего, отбор элементов списка через один, транспонирование списка в обратном порядке.



Рисунок 33. Слайсинг списка – часть 2

Также слайсинг работает и со строкой. Демонстрация на рисунке ниже. Создание строки, отбор всех элементов кроме последнего, транспонирование в обратном порядке — рисунок 34. Все операторы аналогичны вышеописанным.



Рисунок 34. Слайсинг строки

Слайсы позволяют экономить усилия программиста при создании кода.

Операторы условий if / then. Действие оператора показано на рисунке 35. Задаются условия и вывод результата при его выполнении, затем моделируется ситуация его невыполнения — вывода на печать заданного при выполнении условия текста нет, задание текста для распечатки при невыполнении условия оператором else — вывод результата. Оператор else включает ветвление.

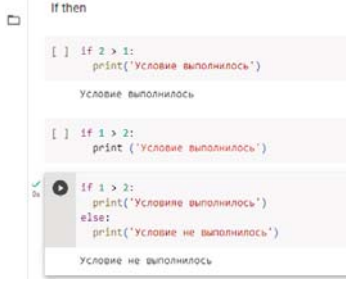


Рисунок 35. Условные операторы

Иллюстрация многоуровневого ветвления.

Задается переменная. Затем ей определяется многоуровневая система условий, как представлено на рисунке 36.

Написание условия — важнейший элемент любого программирования.

List comprehensions (список включений, или правила подстановок) — сокращённая запись циклов.

Создаётся список, содержащий значения с названием nums. Ставится задача, создать новый список с названием new, все элементы которого состоят из элементов первоначального списка, умноженные на два — рисунок 37.

Действия над каждым элементом списка для создания нового — частая операция, для которой был создан функционал list comprehensions. Это работает следующим образом. Открытием квадратных скобок создаётся новый список. При описании исходного списка сразу в начале командной строки в квадратных скобках указывается, что надо делать с исходным списком — рисунок 38.

```
num = 4

[4] print(num)

    4

if num > 11:
  print('num > 11')
elif num > 6:
  print('num больше 6')
else:
  print('число меньше 6')

число меньше 6
```

Рисунок 36. Многоуровневая система условий

```
List comprehensions

[13] nums = [1,2,3,4,5]

[14] nums

     [1, 2, 3, 4, 5]

[20] new = []
     for num in nums:
       new.append(num * 2)

new

[2, 4, 6, 8, 10]
```

Рисунок 37. Создание списка из нового перемножением на два

```
[num * 2 for num in nums]

[2, 4, 6, 8, 10]
```

Рисунок 38. List comprehensions

Так операторы, занимавшие в предыдущем примере три строки, теперь умещаются в одну. Самое главное преимущество этих правил подстановок — они легко интуитивно воспринимаются и составляются.

В правилах подстановок можно использовать условия. Например, при создании подстановок выбираются только те элементы исходного списка nums, которые больше 2 — рисунок 39.

```
[num * 2 for num in nums if num > 2]
[6, 8, 10]
```

Рисунок 39. Операции при подстановке с элементами по заданным условиям

Проверка каждого из элементов списка на определённые условия осуществляется следующей последовательностью команд при подстановке. В данном случае необходимо написать, больше ли элемент 2 — рисунок 40.

```
[True if num > 2 else False for num in nums]
[False, False, True, True, True]
```

Рисунок 40. Условные операции при подстановке

Вложенные подстановки. С помощью этой операции можно создавать списки списков, или матрицы.

Генерация последовательного списка из трёх элементов — рисунок 41.

```
[i for i in range (3)]
[0, 1, 2]
```

Рисунок 41. Последовательная генерация

Генерация одного и того же элемента: единицы — рисунок 42.

```
[1 for i in range (3)]
[1, 1, 1]
```

Рисунок 42. Генерация одинакового элемента

Если в операторе не идёт речь об определённой переменной, в данном случае i, то и обозначать её не надо, а вместо неё можно просто подставить нижнее подчеркивание, как показано на рисунке 43.

```
[1 for _ in range(3)]
[1, 1, 1]
```

Рисунок 43. Нижнее подчеркивание вместо буквенного обозначения

При переборе значений в цикле так делается часто.

Для повторения действия на рисунке 43 три раза запись оператора такая, как на рисунке 44 с выводом результата — матрицы 3×3, таблицы с тремя строками и тремя столбцами.

```
[[1 for _ in range(3)] for _ in range (3)]
[[1, 1, 1], [1, 1, 1], [1, 1, 1]]
```

Рисунок 44. Трёхкратный цикл