

Многоуровневая архитектура управления LLM-ассистентом в предметно-ограниченной образовательной среде

В.М. Абрамян

*Южный федеральный университет,
Институт математики, механики
и компьютерных наук им. И. И. Воровича*

vikabramyan@sfedu.ru



Идея

Метатрон 5.2



Метатрон

От Anastasiia Derzhanskaia

✓ Использование модели, рекомендованной создателем: GPT-5.2 Instant

GPT-ассистент для изучения основ программирования на языке Python

Вывести список тем

Метатрон 5.2

Список тем курса (см.):

1. Тема 1 — Вывод и ввод данных. Переменные.

2. Тема 2 — Арифметические операции в Python

3. Тема 3 — Условный оператор в Python

Выберите тему, которую хотите изучить.

📄

👍

👎

...

Метатрон 5.2

Вывести список тем

Метатрон 5.2

Спросите ChatGPT

Метатрон 5.2

Выберите тему, которую хотите изучить.

📄

👍

👎

...

Раздел 1. Целые числа

1.1. Числовые данные в Python

В предыдущей теме мы работали в основном со строками. Это естественно, потому что команда `input()` по умолчанию считывает именно текст. Даже если пользователь введёт с клавиатуры число, Python сначала воспримет его как строку.

Однако в программировании очень часто приходится работать не с текстом, а с числами. Например, программа может складывать значения, вычислять площадь фигуры, находить среднее арифметическое,

Метатрон 5.2

Поделиться

Метатрон 5.2

В этой программе переменные `num1` и `num2` содержат целые числа, а переменная `num3` получает результат их сложения.

Теперь сравним это с другой программой:

Python

```
num1 = '7'
num2 = '10'
num3 = num1 + num2

print(num3)
```

Запустить

Вывод программы:

710

На экране мы видим уже "склейку" этих чисел, потому что сложение двух строк склеивает их вместе.

Запомним:

Метатрон 5.2

Поделиться

Метатрон 5.2

Если вы пишете:

Python

```
print(3 + 5)
```

Запустить

то результат будет:

8

Потому что 3 и 5 — это числа, записанные без кавычек, и выполняется обычное сложение.

А вот если написать так:

Python

```
print('3' + '5')
```

Запустить

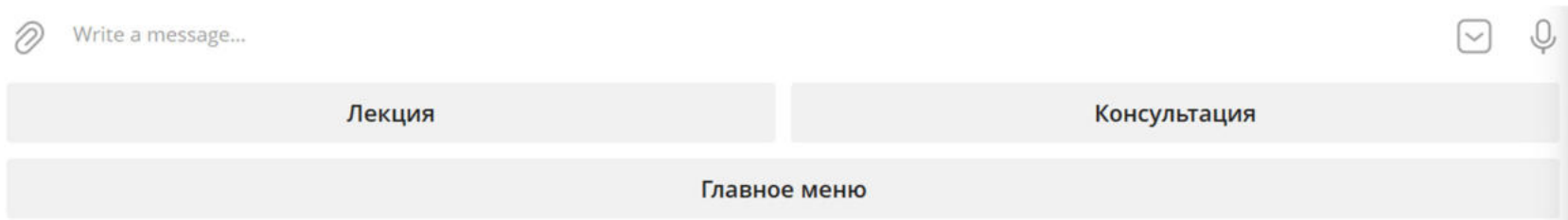
3 / 15

Проблемы

- GPT-ассистент «слишком умный», чтобы просто воспроизводить текст лекции.
- Со сменой модели кардинально меняется поведение.
- Больше курс — дороже контент.

Решение

Режимы



- Лекция:
 - Пошаговые объяснения темы.
 - Вопросы в рамках лекции.
- Тест:
 - Последовательные вопросы.
 - Изменение формулировки по запросу студента.
- Консультация:
 - Вопросы по ранее изученному материалу.
 - Ступенчатая помощь при решении задач.



Лекция

Metatron bot

Главное меню 17:33 ✓

Главное меню.

Выберите режим: 17:33

Лекция 17:33 ✓

Режим лекции.

Выберите лекцию или тест: 17:33

Лекция 3 17:33 ✓

Раздел 1. Условный оператор if-else

1.1. Зачем нужен условный оператор

В предыдущих темах мы научились вводить данные, хранить их в переменных и выполнять вычисления. Однако программа должна уметь не только обрабатывать данные, но и принимать решения.

Во многих задачах дальнейшие действия зависят от введённых значений. Например, программа может проверить возраст пользователя, сравнить два числа, определить, выполнено ли условие, и в зависимости от результата выполнить разные команды.

Такое поведение называется ветвлением.

Ветвление — это проверка условия и выбор дальнейших действий в зависимости от результата этой проверки.

В Python результат проверки условия всегда сводится к одному из двух значений:

Дальше

План темы

Тест по теме

Консультация

Назад

Главное меню

Metatron bot

Для проверки условий в Python используется ключевое слово if. 17:33

Если есть вопросы, задайте их. Если нет — скажите, и мы продолжим. 17:33

Дальше 17:35 ✓

1.2. Как работает if

Рассмотрим пример программы: 17:35

```
python
answer = input('Какой язык программирования мы изучаем?')
if answer == 'Python':
    print('Верно! Мы изучаем Python.')
    print('Три недели занятий не прошли даром.')
```

17:35

Эта программа просит пользователя ввести текст, а затем проверяет результат ввода. Если введённая строка равна 'Python', выполняются две команды print().

Вывод программы при подходящем вводе: 17:35

```
copy
Верно! Мы изучаем Python.
Три недели занятий не прошли даром.
```

17:35

После строки с инструкцией if ставится двоеточие :. Оно показывает Python, что дальше начинается блок кода. 17:35

Если есть вопросы, задайте их. Если нет — скажите, и мы продолжим. 17:35

Дальше

План темы

Тест по теме

Консультация

Назад

Главное меню



Лекция

Metatron
bot

Верно! Мы изучаем Python.
Три недели занятий не прошли даром.

После строки с инструкцией if ставится двоеточие :. Оно показывает Python, что дальше начинается блок кода.

Если есть вопросы, задайте их. Если нет — скажите, и мы продолжим.

План темы 17:39 ✓

План темы:

- 1.1 Зачем нужен условный оператор
- 1.2 Как работает if
- 1.3 Блок кода и отступы
- 1.4 Конструкция if-else
- 2.1 Присваивание и сравнение
- 2.2 Основные операторы сравнения
- 2.3 Цепочки сравнений
- 2.4 Транзитивность сравнения
- 3.1 Зачем нужны логические операторы
- 3.2 Оператор and
- 3.3 Оператор or
- 3.4 Оператор not
- 3.5 Совместное использование and и or
- 3.6 Приоритет логических операторов
- 3.7 Порядок вычисления в составных условиях
- 3.8 Укороченная схема вычисления
- 4.1 Вложенный условный оператор
- 4.2 Каскадный условный оператор if-elif-else
- 4.3 Почему if-elif-else удобнее
- 4.4 Блок else необязателен
- 5.1 Отсутствие двоеточия после условия
- 5.2 Использование = вместо ==
- 5.3 Отсутствие отступа внутри блока if
- 5.4 if и else находятся не на одном уровне
- 5.5 Неправильное использование операторов сравнения
- 5.6 Неполная запись составного условия
- 5.7 Неправильное использование or при сравнении со строками
- 6.1 Повторение (кратко)

Давай продолжим лекцию с пункта 3.2 17:42 ✓

Metatron
bot

Metatron
bot

Давай продолжим лекцию с пункта 3.2 17:42 ✓

3.2. Оператор and

Оператор and объединяет несколько условий и возвращает истину только в том случае, если все эти условия истинны.

Рассмотрим пример:

```
python
age = int(input('Сколько вам лет?: '))
grade = int(input('В каком классе вы учитесь?: '))
if age >= 12 and grade >= 7:
    print('Доступ разрешен.')
else:
    print('Доступ запрещен.')
```

В этой программе доступ разрешается только тогда, когда одновременно выполняются два условия:

- возраст не меньше 12;
- класс не ниже 7.

Оператор and может объединять и больше двух условий:

```
python
age = int(input('Сколько вам лет?: '))
grade = int(input('В каком классе вы учитесь?: '))
city = input('В каком городе вы живете?: ')
if age >= 12 and grade >= 7 and city == 'Москва':
    print('Доступ разрешен.')
else:
    print('Доступ запрещен.')
```

Здесь доступ разрешается только в том случае, если истинны все три условия.

Таблица истинности для оператора and:

a	b	a and b
True	True	True
True	False	False
False	True	False
False	False	False

Metatron
bot

Write a message...



Лекция

```
1 [TOPIC START: 3]
2 # Тема 3. Условный оператор в Python
3
4 [TOPIC MAP START: 3]
5 TOPIC_ID: TOPIC_3
6 TOPIC_TITLE: Тема 3. Условный оператор в Python
7
8 [SECTION 1]
9 1.1 -> BLOCK_3_1_1 -> Зачем нужен условный оператор
10 1.2 -> BLOCK_3_1_2 -> Как работает if
11 1.3 -> BLOCK_3_1_3 -> Блок кода и отступы
12 1.4 -> BLOCK_3_1_4 -> Конструкция if-else
13
14 [SECTION 2]
15 2.1 -> BLOCK_3_2_1 -> Присваивание и сравнение
16 2.2 -> BLOCK_3_2_2 -> Основные операторы сравнения
17 2.3 -> BLOCK_3_2_3 -> Цепочки сравнений
18 2.4 -> BLOCK_3_2_4 -> Транзитивность сравнения
19
20 [SECTION 3]
21 3.1 -> BLOCK_3_3_1 -> Зачем нужны логические операторы
22 3.2 -> BLOCK_3_3_2 -> Оператор and
23 3.3 -> BLOCK_3_3_3 -> Оператор or
24 3.4 -> BLOCK_3_3_4 -> Оператор not
25 3.5 -> BLOCK_3_3_5 -> Совместное использование and и or
26 3.6 -> BLOCK_3_3_6 -> Приоритет логических операторов
27 3.7 -> BLOCK_3_3_7 -> Порядок вычисления в составных условиях
28 3.8 -> BLOCK_3_3_8 -> Укороченная схема вычисления
29
30 [SECTION 4]
31 4.1 -> BLOCK_3_4_1 -> Вложенный условный оператор
32 4.2 -> BLOCK_3_4_2 -> Каскадный условный оператор if-elif-else
33 4.3 -> BLOCK_3_4_3 -> Почему if-elif-else удобнее
34 4.4 -> BLOCK_3_4_4 -> Блок else необязателен
35
36 [SECTION 5]
37 5.1 -> BLOCK_3_5_1 -> Отсутствие двоеточия после условия
38 5.2 -> BLOCK_3_5_2 -> Использование = вместо ==
39 5.3 -> BLOCK_3_5_3 -> Отсутствие отступа внутри блока if
40 5.4 -> BLOCK_3_5_4 -> if и else находятся не на одном уровне
41 5.5 -> BLOCK_3_5_5 -> Неправильное использование операторов сравнения
42 5.6 -> BLOCK_3_5_6 -> Неполная запись составного условия
43 5.7 -> BLOCK_3_5_7 -> Неправильное использование or при сравнении со строками
44
45 [SECTION 6]
46 6.1 -> BLOCK_3_6_1 -> Повторение (кратко)
47 [TOPIC MAP END: 3]
```

```
98 [BLOCK START: BLOCK_3_1_3]
99 [BLOCK NUMBER: 1.3]
100 [BLOCK TITLE: Блок кода и отступы]
101 [BLOCK TITLE: Блок кода и отступы]
102 ### 1.3. Блок кода и отступы
103
104 **Блок кода** – это несколько строк программы, которые
относятся к одной инструкции и выполняются как единое целое.
105
106 В Python блоки кода выделяются с помощью **отступов**. Это не
просто вопрос аккуратного оформления. Для Python отступы
являются частью синтаксиса, поэтому без них программа либо не
запустится, либо будет работать неправильно.
107
108 В других языках программирования границы блока могут
обозначаться специальными символами, например фигурными
скобками. В Python для этого используются именно отступы. По
ним интерпретатор определяет, какие строки относятся к
условному оператору, а какие уже находятся за его пределами.
109
110 **Отступ** – это небольшое смещение строки вправо. Обычно в
начале такой строки стоят пробелы.
111
112 Схема записи условного оператора выглядит так:
113
114 ```python
115 if условие:
116     блок кода
117 ```
118
119 Все строки с отступом после if считаются частью одного блока.
Если условие истинно, выполняется весь этот блок. Как только
Python встречает строку без нужного отступа, он считает, что
блок закончился. Если условие ложно, весь блок после if
пропускается.
120
121 Поэтому отступы в Python критически важны: они не просто
делают код более читаемым, но напрямую влияют на работу
программы.
122
123 По соглашению PEP 8 для одного уровня отступа используют 4
пробела.
124
125 [BLOCK END: BLOCK_3_1_3]
126
127 [BLOCK START: BLOCK_3_1_4]
128 [BLOCK NUMBER: 1.4]
```



Тест

Metatron
bot

False | True | False |
True | False | False |
True | True | True |

Из таблицы видно: выражение с оператором and истинно только тогда, когда истинны оба выражения. 17:42

Если есть вопросы, задайте их. Если нет — нажмите «Дальше». 17:42

Тест по теме 17:48 ✓

Тест начался. 17:48

Что покажет приведенный ниже код: 17:48

```
python
language = 'English'

if language == 'Русский':
    print('Привет!')

if language == 'English':
    print('Hi!')
```

17:48

1. Привет!
2. Hi!
3. Привет
Hi!
4. Hi!
Привет!
5. Не покажет ничего
6. Произойдет ошибка 17:48

Вопрос 1 из 8. Ответьте номером варианта. 17:48

Write a message...

Назад Главное меню



Консультация

Элементы контроля:

- Промпт.
- Постобработка ответа модели.

Консультация

Промпт:

- Перечислены пройденные темы и запрещены все остальные.
- Зафиксирован принцип минимально достаточной помощи и задано поэтапное усиление поддержки.
- Полное решение запрещено.



Консультация

Этапы проверки при постобработке:

- Выход за рамки пройденного материала:
 - Маркеры непройденных тем.
- Слишком сильная помощь.
- Не является ли ответ уже готовым решением:
 - По фразам.
 - По объему fenced-блоков.



Безопасность системы

- Валидация контента при запуске.
- Разделение состояния пользователя.
- Whitelist пользователей.

Вывод

- В предметно ограниченных курсах технически возможно построение LLM-ассистентов, которые устойчиво работают в пределах хронологии и содержания курса.
- Полученное решение остается модульным и переносимым, что позволяет адаптировать его к другим платформам и курсам.