

Язык программирования Python

Лекция №1

ОСНОВЫ ЯЗЫКА PYTHON

Лектор Аксентьев Артем Алексеевич

Приветствие, очень рад Вас здесь видеть и тд

Что изучим на курсе

- **Как программировать оптимально**
- **Распространенные библиотеки**
- **Хранение и обмен проектами**
- **Документирование кода**

Данный курс поможет Вам научиться:

находить оптимальные алгоритмы для решения разнообразных задач

Использовать распространенные библиотеки, как для работы в сфере ML, так и для других сфер

Хранить и обмениваться проектами при помощи системы контроля версий Git

Писать чистый код, который могут читать, понимать и использовать другие люди

ОБЫЧНЫЙ ПРОЦЕСС ОБУЧЕНИЯ



ШАГ 1

Много теории

ШАГ 2

Постановка задачи

ШАГ 3

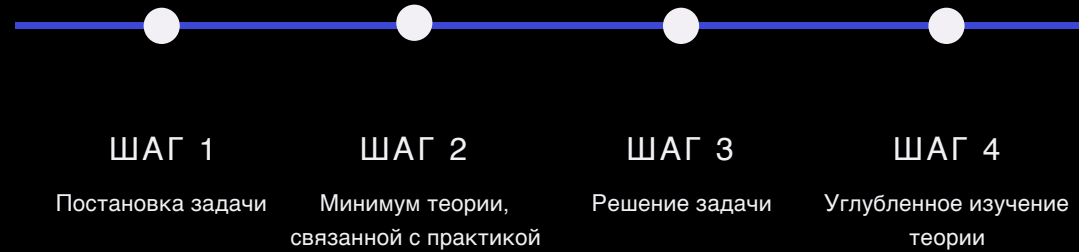
Решение задачи

ШАГ 4

Возврат к теории

Мой курс немного отличается от привычных Вам лекций, семинаров, лабораторных работ. В процессе обучения обычно дают много теории, если повезет то лабораторные будут выполняться по мере прохождения теории.

ОБУЧЕНИЕ НА ДАННОМ КУРСЕ



В моем курсе я постарался дать минимальное количество теории, которую сразу понятно куда, как и зачем применять. Естественно в таком подходе есть небольшой минус, некоторые вещи могут быть упущены из виду, однако появится понимание того как программировать.

Возникла проблема?

- Изучите документацию, литературу, код модуля
- Четко сформулируйте вопрос
- Ищите в гугле
- Задайте вопрос на профильном сайте или форуме

Что же делать в том случае, если знаний полученных на курсе не хватило?

В первую очередь стоит прочитать документацию самого языка, либо же модуля, который Вы используете. Возможно Вам придется использовать дополнительную литературу или даже открыть код модуля, для понимания процессов, производимых в нем.

Если проблема не решилась, необходимо как можно более четко сформулировать вопрос и пойти искать в гугле.

И лишь после этого стоит обратиться на профильный сайт или же форум.

Сферы программирования

DESKTOP

Windows, Linux,
MacOS
C++, C#, SWIFT

MOBILE

Android, iOS
C#, SWIFT, JAVA,
Kotlin

WEB

Сайты и
приложения
HTML, CSS, PHP,
Python

EMBEDDED

IoT, драйвера
Assembler, C, C++

Задать вопрос: Кто знаком с программированием?

Отлично, я думаю, что вы сможете назвать четыре сферы или типа программирования

Десктоп: это разработка под персональные компьютеры. Практически любые программы запускаемые непосредственно на вашем ноутбуке/компьютере можно отнести к этой сфере. графические библиотеки (Например QT, WindowsForms)

В основном применяются высокоуровневые языки и какие-либо

Мобильная разработка - разработка под мобильные устройства (планшеты, компьютеры, умные часы, умные телевизоры) Эта область очень похожа на десктопное программирование, так же высокоуровневые языки и Фреймворк для работы с интерфейсом

Web - любые сайты, некоторые мобильные приложения, все то что использует технологию интернета. Практически всегда реализуется в виде клиент-серверного приложения, где основные операции выполняются не на стороне клиента, а на сервере

Встраиваемые системы - сюда можно отнести программирование элементов умного дома, вещей, которые общаются через интернет, в отличии от Web и мобильной разработки, чаще всего отсутствует графический интерфейс, и обработкой данных занимаются другие устройства. В этой сфере программирования требуется знание "железной" части устройства, умение использовать более низкий уровень программирования.

Сферы программирования

GAMEDEV

Игры, физика,
графика
C++, C#

BUSINESS

Автоматизация
бизнеса
1C, SAP

SCIENCE

Машинное обучение,
моделирование систем
R, Python

Геймдев - разработка игр, по сути мало чем отличается от разработки для десктопа или мобильных устройств, однако, вместо готовых фреймворков, чаще всего логика, физика и графика программируют вручную

Автоматизация бизнеса - автоматизация рутинных бизнес процессов, например подготовка документов для бухгалтерии или же автоматизация процессов продажи и производства товара.

Научная сфера - можно сказать, что эта сфера может быть включена в любые другие. В эту сферу можно отнести методы машинного обучения и моделирования систем.

Естественно все перечисленные сферы некая условность, нельзя сказать, что какое-либо приложение подходит под какую либо сферу и не подходит под другие.

Из чего же состоит программирование сегодня?

Программирование сегодня



АЛГОРИТМЫ

Умение оптимизировать решение



ПРЕДМЕТНАЯ ОБЛАСТЬ



УПРАВЛЕНИЕ

Распределение заданий, сил и т.д.



ИНСТРУМЕНТЫ

Языки программирования,
фреймворки, библиотеки

Это умение составить правильный и оптимальный алгоритм

Знание той области, для которой эта программа пишется, например автоматизировать заполнение какой-либо бумаги без понимания того откуда берутся цифры невозможно

Немаловажным является умение разделить задачу на подзадачи и определить приоритет каждой задачи.

И наконец выбор оптимального инструмента, фреймворка или библиотеки.

Что изучим сегодня

- Алгоритмы, данные, оценку эффективности алгоритмов
- Базовый синтаксис языка Python
- Базовые приемы хранения и обмена кода
- Правила написания "чистого" кода

КОМПЬЮТЕРНОЕ МЫШЛЕНИЕ



ВХОДНЫЕ
ДАННЫЕ

АЛГОРИТМЫ

ВЫХОДНЫЕ
ДАННЫЕ

Как думает компьютер?

Хранение цифр в памяти
компьютера

123

Хранение цифр в памяти компьютера

123

$$1 * 100 + 2 * 10 + 3 * 1$$

Хранение цифр в памяти компьютера

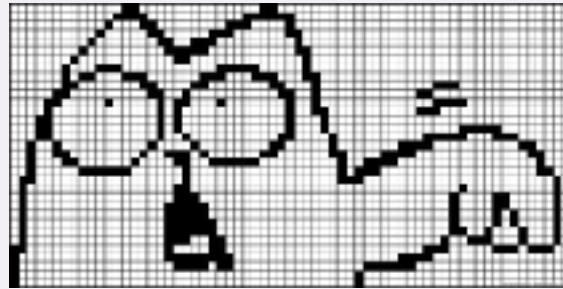
101010

$$1 * 32 + 0 * 16 + 1 * 8 + 0 * 4 + 1 * 2 + 0 * 1$$

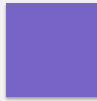
Хранение букв в памяти компьютера

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G
0	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0G
1	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	1G
2	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	2G
3	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	3G
4	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	4G
5	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	5G
6	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	6G
7	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	7G
8	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	8G
9	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	9G

Хранение цвета в памяти компьютера



Хранение цвета в памяти компьютера



120



100



198

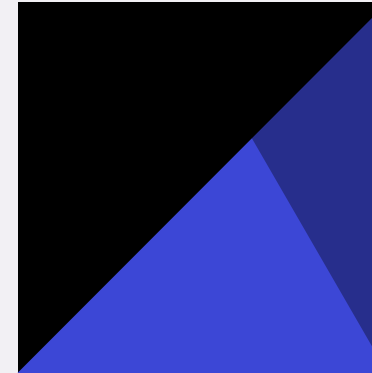


Задача о мышах

У Вас есть 1000 пробирок, 999 из них содержат лекарства, а одна яд. Лекарство полностью безвредное. Даже капля яда приводит к смерти через 12 часов после приема.

Так же у Вас в наличии 10 мышей.

Ваша задача за минимальное время найти пробирку с ядом.

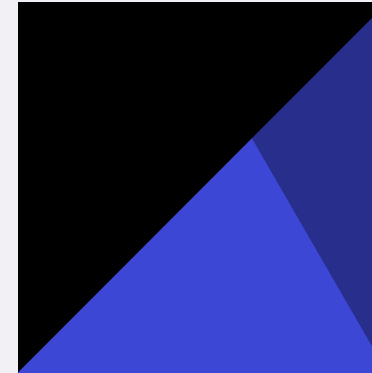


Давайте попробуем решить такую задачу

Задача о мышах

Подсказки

- Рационально ли выбирать какие-то отдельные пробирки или нужно протестировать всё?
- Есть ли в условии задачи лимит по количеству введённого лекарства?
- Можно ли использовать одно животное несколько раз или дать ему лекарства из нескольких пробирок?

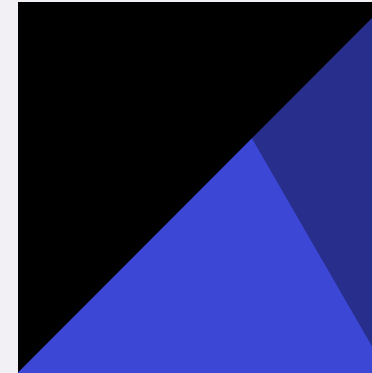


Задача о мышах

Решение

Пронумеруем пробирки от 1 до 1000. Найдем **битовую глубину** этой нумерации. $1000 = 1111101000$, битовая глубина равна 10 цифрам

1 - 00000000001
2 - 00000000010
....
1000 - 1111101000





Набор действий, которые преобразуют входные данные в выходные это и есть алгоритм.

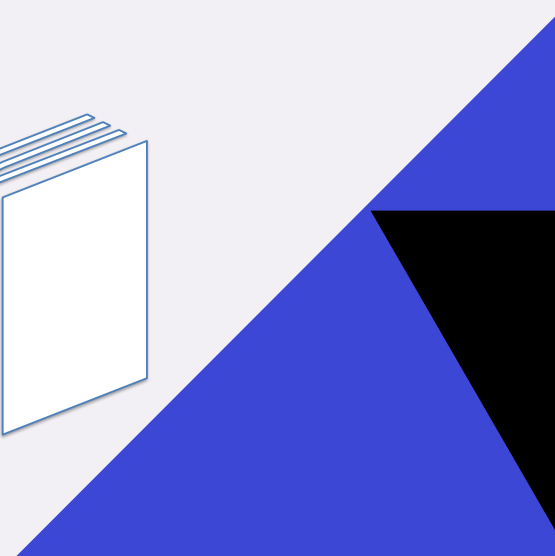
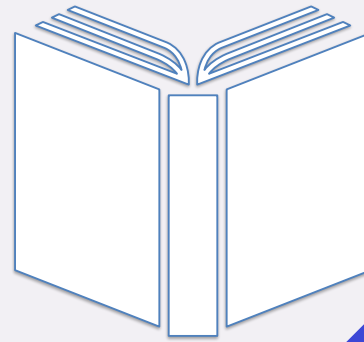
Хороший алгоритм обладает несколькими свойствами:
Определенность

Массовость

Результативности

ПОИСК ЭЛЕМЕНТА

В словаре найти слово
"Программирование"



Алгоритмы поиска в отсортированном массиве

ЛИНЕЙНЫЙ

Проверяем каждый
элемент

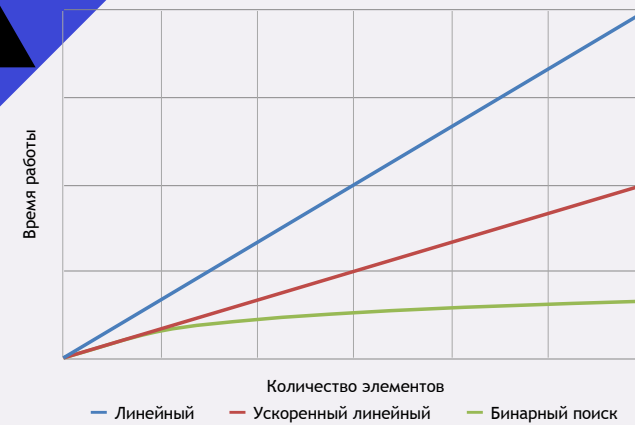
"УСКОРЕННЫЙ"

Проверяем каждый
второй элемент

БИНАРНЫЙ

Делим массив
пополам и решаем
задачу с половиной
массива

ЭФФЕКТИВНОСТЬ АЛГОРИТМА



ПРИМЕЧАНИЕ

В информатике для оценки эффективности используется О-нотация

НАЧНЕМ ПРОГРАММИРОВАТЬ

ЕСТЕСТВЕННО НА ЯЗЫКЕ PYTHON

СИНТАКСИС

ИНСТРУКЦИИ

Каждая строка - одна инструкция

КОММЕНТАРИИ

Начинаются с символа #

ИНТЕРАКТИВНЫЙ РЕЖИМ

Можно выполнять программу по частям

Каждая строка одна инструкция. Но иногда можно записывать несколько инструкций в одну строку, разделив их ;

В Python существует два режима выполнения программы:

- Интерактивный
- Файловый

Комментарии используются для пояснения малопонятных частей кода



```
1.print("Hello, world!")
2.# Hello, world!

>>>print("Hello, world!")
Hello, world!
```

```
a = 10
b = 'Hello world'
c = 1.0
b = 12
```

```
Input/output
>>>name = input("Введите своё имя: ")
Введите своё имя: Иван
>>>print("Привет, ", name)

>>>name, b = 5, 10
>>>print(name + b)
15
>>>name, b = b, name
>>>print(name, b)
10 5
```

```
a = 10
b = 'Hello world'
c = 1.0
b = 12
```

ПЕРЕМЕННЫЕ

A

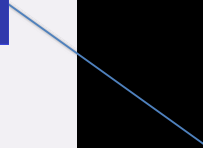
B

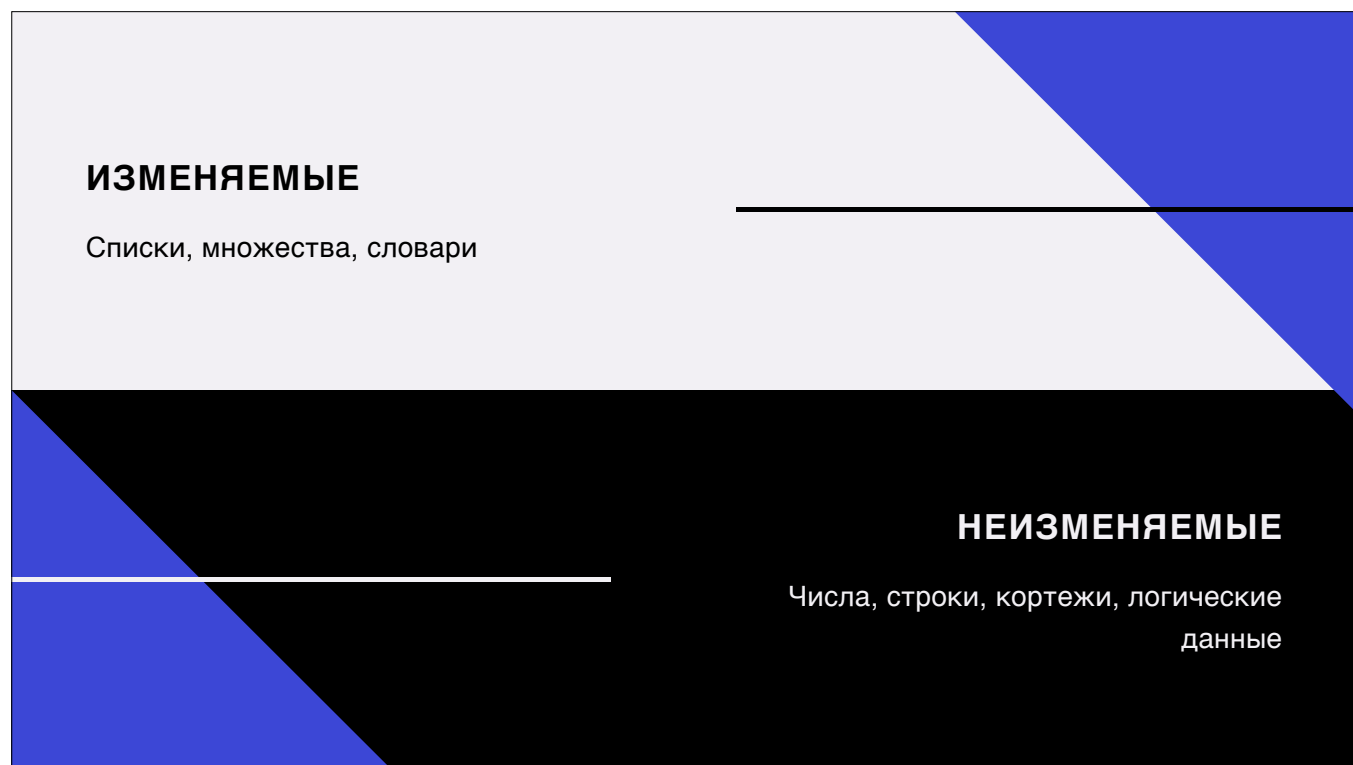
ДАННЫЕ

10

"Hello"

12





Все данные делятся на два типа

Сегодня мы рассмотрим списки, числа, строки и логические данные

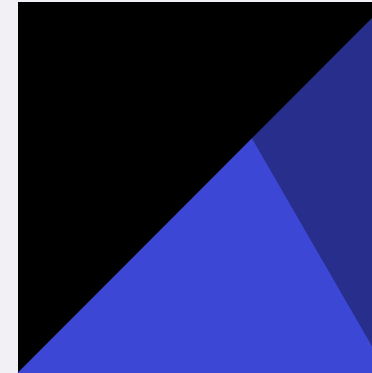
Целые числа

Не имеют ограничений

Не задумывайтесь об этом

**Поддерживают стандартные
математические операции**

$x + y$, $x - y$, x / y , $x // y$, $x \% y$, $\text{abs}(x)$, $x^{**}y$



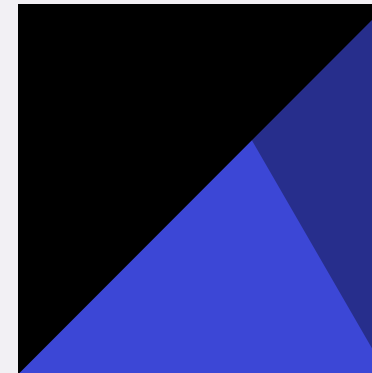
Вещественные числа

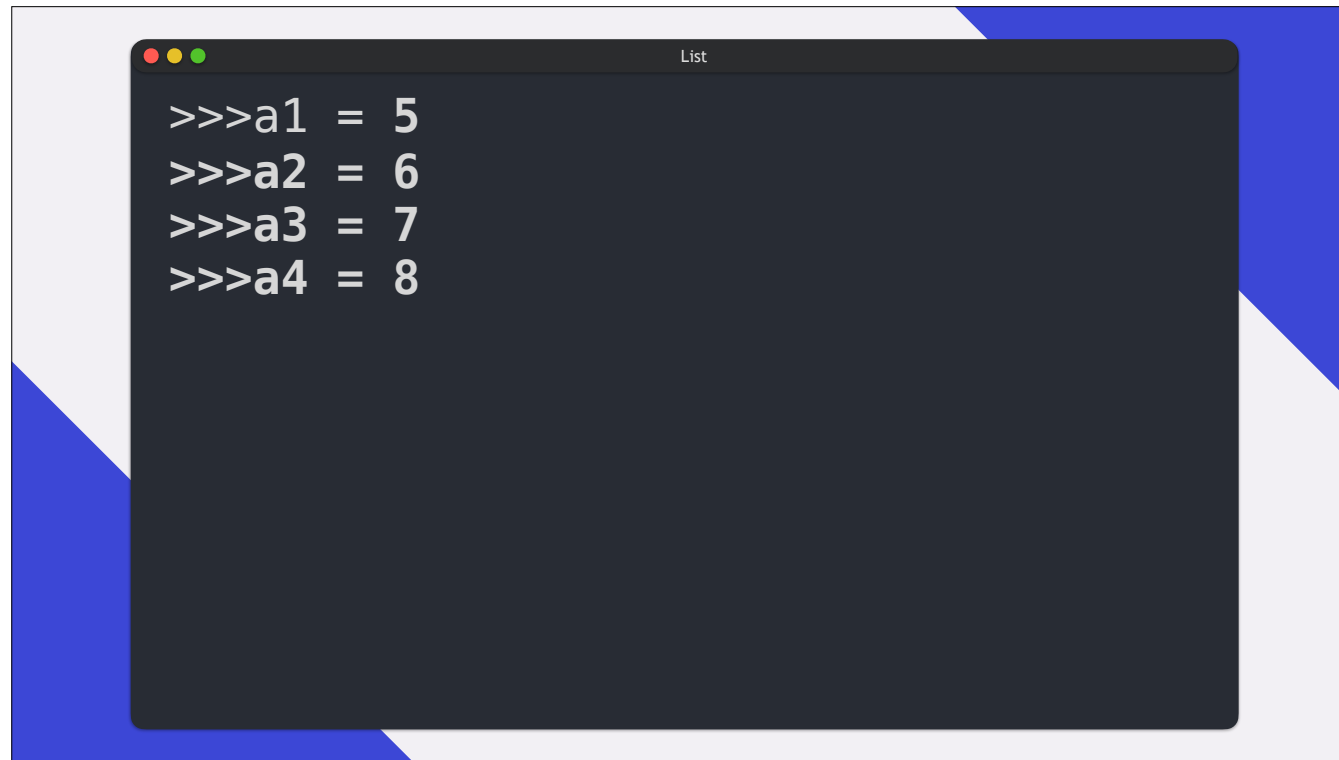
Те же операции, что и над целыми

Имеют некоторые ограничения на
размер

Разделителем дробной части
является точка

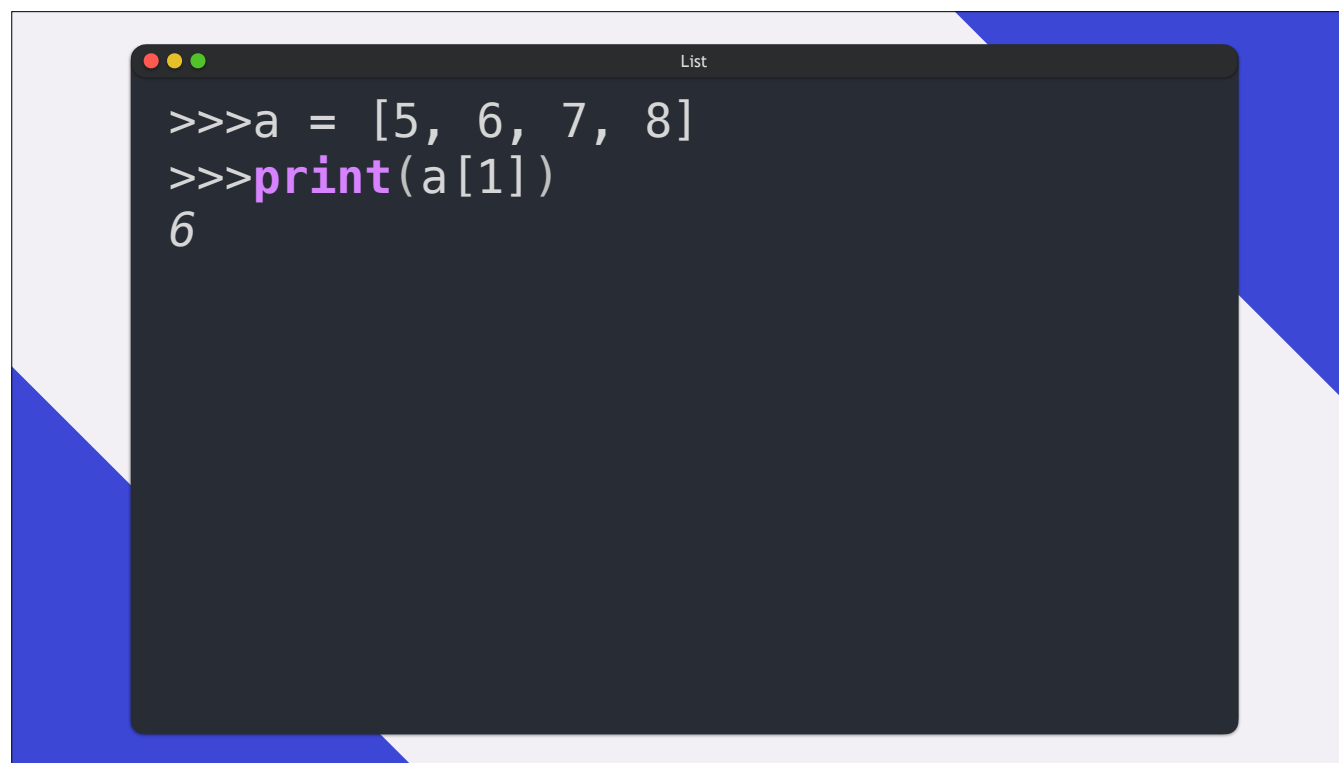
$0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1$
 $+ 0.1 + 0.1 = 0.999999999999$





Что не так? Необходимо создавать каждый раз необходимо новую переменную, что глупо

Кроме того мы не сможем удобно обращаться к ним при необходимости

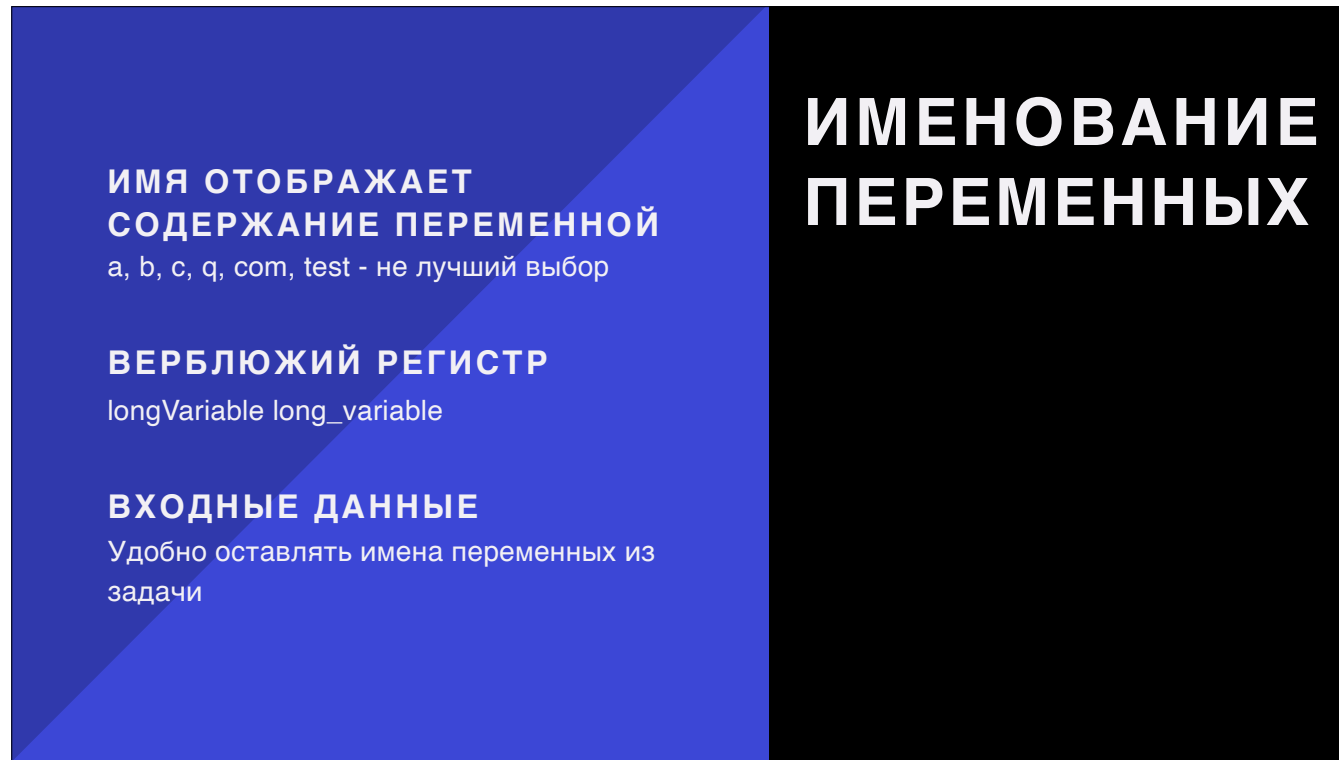
A terminal window with a dark background and a title bar that says "List". The window contains three lines of text: a Python list assignment, a print statement, and the output of that statement. The code is written in a monospaced font, with the word "print" in a light blue color. The output "6" is on a new line.

```
>>>a = [5, 6, 7, 8]
>>>print(a[1])
6
```

Все эти две проблемы решают списки или массивы

Списки

- Нумеруются с нуля
- Можно обращаться по индексу
- Можем изменять элементы
- Вывести список с помощью print



Однобуквенные имена иногда могут использоваться в олимпиадных задачах, где важна скорость написания кода и с большой вероятностью этот код не будет никто смотреть

Стиль именования переменных напрямую не оговорен в стандартах Python, но я предлагаю придерживаться следующих нотаций

CODESTYLE

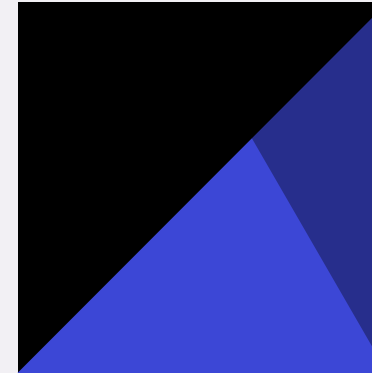
Максимальная длина строки 79 символов

Можно отобразить в любом текстовом редакторе
и в нескольких вкладках

Строго один пробел между операторами

`a = 0` `a = 0`
`long = 0` `long = 0`

**Особые случаи не настолько
особые, чтобы нарушать правила.**



КОММЕНТАРИИ

Не противоречат коду

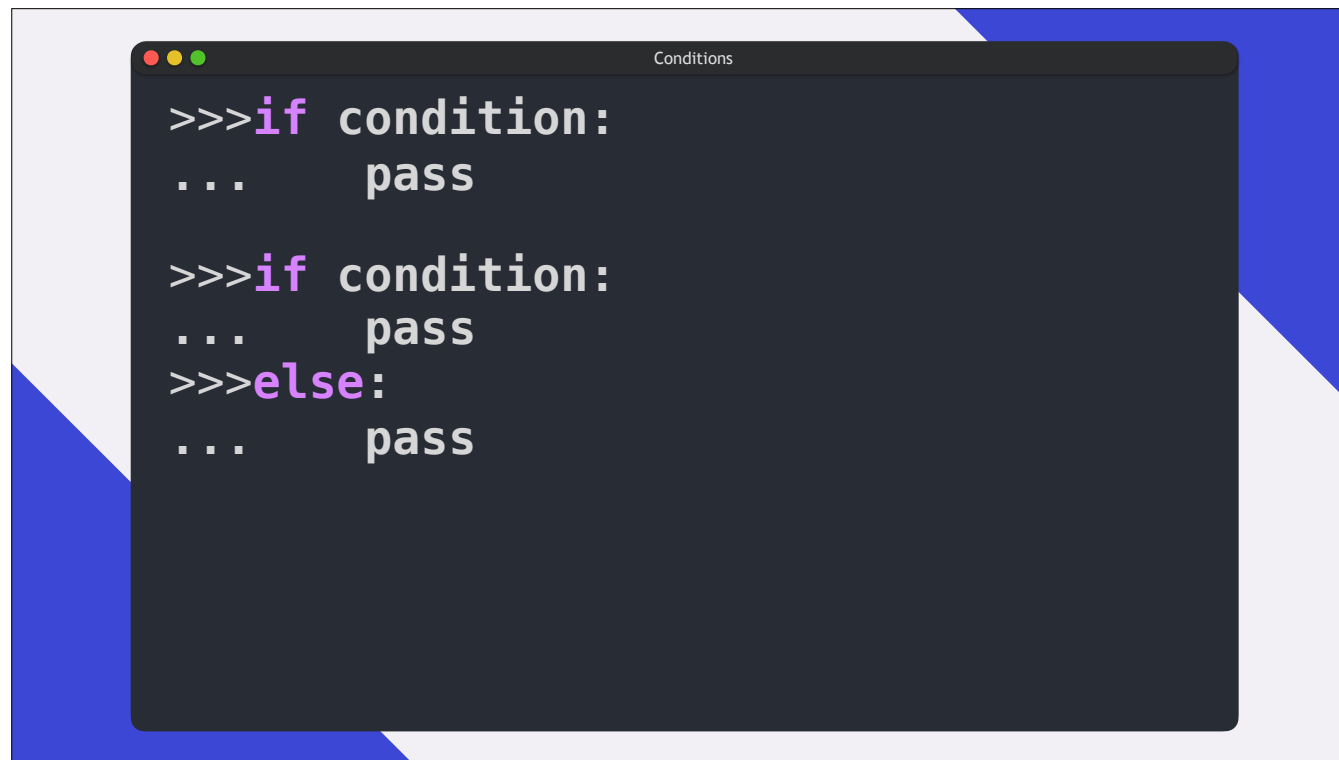
Изменил код, измени и комментарий

Законченные предложения

Кратко, но емко

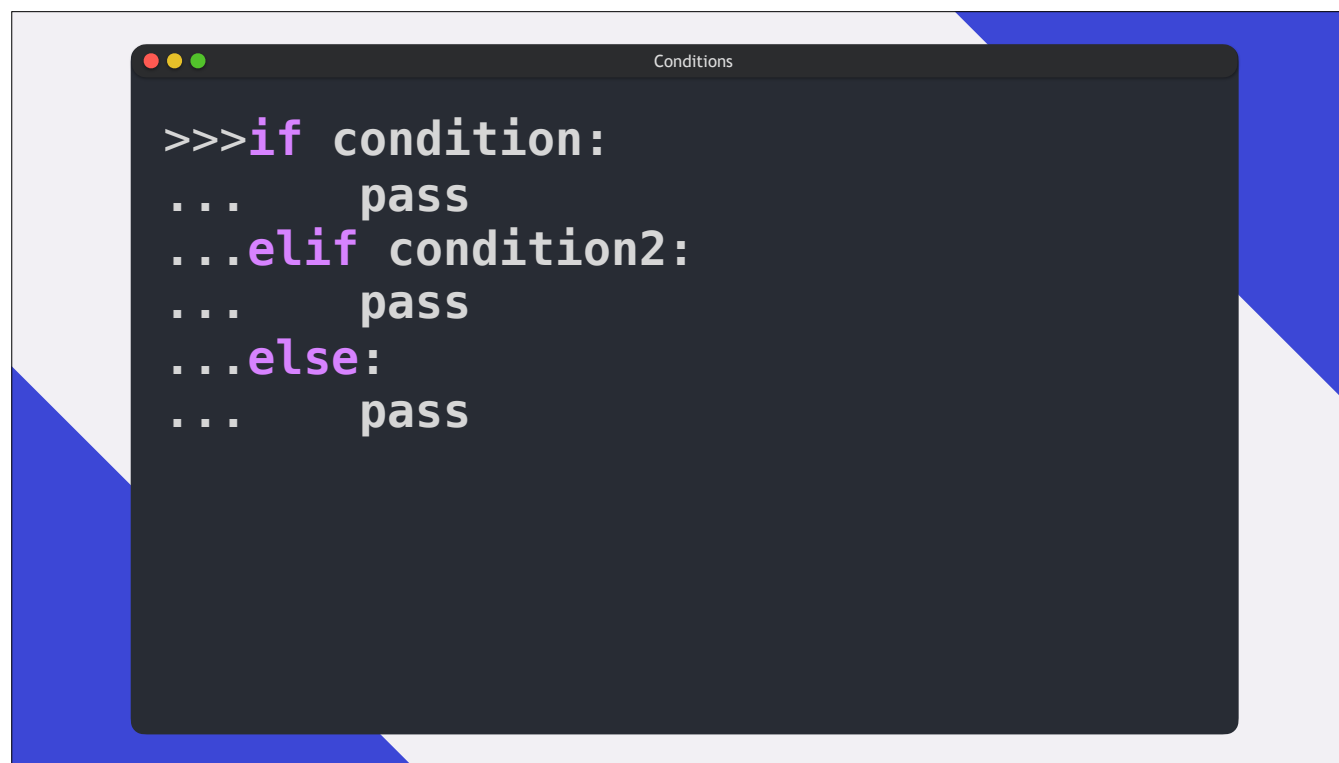
Комментарии и код по возможности на английском

Самое плохое, когда комментарий противоречит коду
Если одно предложение, то можно опустить точку

A code editor window titled "Conditions" with a dark background and light-colored text. The window contains two Python code snippets. The first snippet shows an if statement with a condition and a pass statement. The second snippet shows an if-else statement with conditions and pass statements. The code is formatted with indentation and color-coding for keywords.

```
>>>if condition:
...     pass

>>>if condition:
...     pass
>>>else:
...     pass
```



```
>>> if condition:
...     pass
... elif condition2:
...     pass
... else:
...     pass
```

Логические ошибки

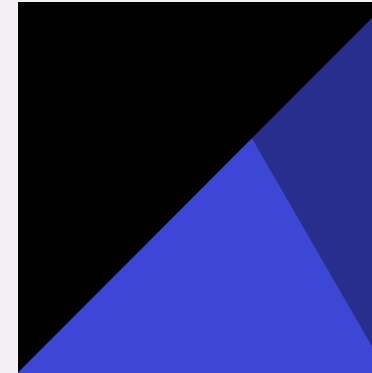
Ошибки в логике работы программы.
Проверка данных на некоторых шагах
исполнения

Синтаксические ошибки

Ошибки в коде. Интерпретатор
покажет их

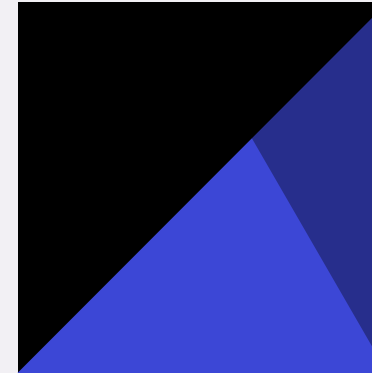
Дзен Python

- Красивое лучше, чем уродливое.
- Явное лучше, чем неявное.
- Простое лучше, чем сложное.
- Сложное лучше, чем запутанное.
- Плоское лучше, чем вложенное.
- Разреженное лучше, чем плотное.
- Читаемость имеет значение.
- Особые случаи не настолько особые, чтобы нарушать правила.
- При этом практичность важнее безупречности.
- Ошибки никогда не должны замалчиваться.
- Если не замалчиваются явно.



Дзен Python

- Встретив двусмысленность, отбрось искушение угадать.
- Должен существовать один — и, желательно, только один — очевидный способ сделать это.
- Хотя он поначалу может быть и не очевиден, если вы не голландец.
- Сейчас лучше, чем никогда.
- Хотя никогда зачастую лучше, чем прямо сейчас.
- Если реализацию сложно объяснить — идея плоха.
- Если реализацию легко объяснить — идея, возможно, хороша.
- Пространства имён — отличная вещь! Давайте будем делать их больше!



КОМАНДА РАЗРАБОТЧИКОВ ИГРЫ



Геймдизайнер



Программист



Художник



Звукорежиссер



Тестировщик

VCS

Работа в команде

Совместный труд он объединяет

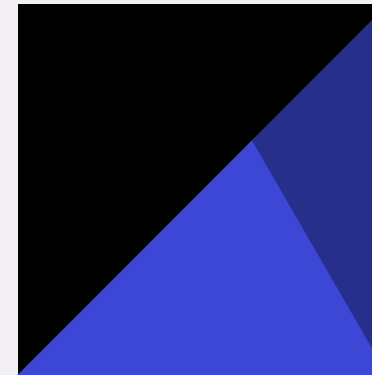
Контроль версий

Хранение этапов работы

Ветвление

Несколько версий программы

Контроль доступа



Области в *git*

Рабочая область

Та область где мы работаем

Подготовленные файлы (Staged)

Зафиксированные, но не сохраненные изменения

Репозиторий

Сохраненный снимок файлов

Git хранит данные в виде неких снимков файловой системы, которые называется коммитами.

У каждого комета есть уникальный идентификатор, по которому мы можем обратиться к ней.

Так же git хранит ссылку на самый последний коммит

Статус данных в Git

- Не отслеживается (untracked)
- Подготовлен (staged)
- Изменен (modified)
- Без изменений

Что не следует помещать в Git?

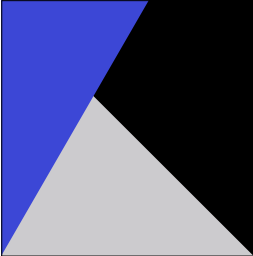
- **Файлы с конфиденциальной информацией**
- **Большие бинарные файлы**
- **Файлы специфичные для OS/IDE**
- **Логи, файлы создаваемые в процессе компиляции**

Советы для по коммитам

- Не бойтесь делать их слишком много
- Старайтесь, чтобы каждый коммит содержал одно изменение
- Оставляйте сообщение для коммита
- Не отправляйте на сервер кучу коммитов

Задания для самостоятельного решения

ЕСТЕСТВЕННО НА ЯЗЫКЕ PYTHON



Советы для выполнения заданий

- **Подумайте над алгоритмом**

Проанализируйте задачу и подумайте, как её решить

- **Попробуйте оптимизировать**

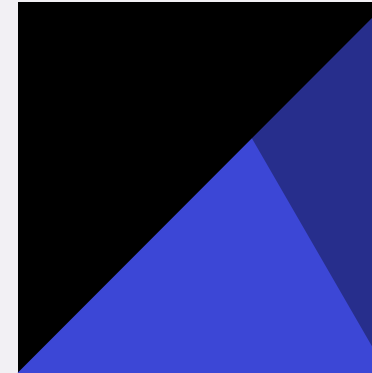
- **Продумайте все возможные варианты**

- **Не списывайте**

А если списали, разберитесь в коде

Калькулятор

Напишите простое приложение, которое общается с пользователем, предлагая ему ввести два операнда и одно из арифметических действий, после чего выведите результат. Программа должна выполняться до тех пор пока пользователь в любой момент не введет "quit"



Оценка решений

КОРРЕКТНОСТЬ

Решение должно
выполнять
поставленную задачу.
Не забывайте о
граничных случаях

ВРЕМЯ

Оптимальное
время работы

ПАМЯТЬ

Не должна
занимать лишнюю
память

CODESTYLE

Должно
соответствовать
стандартам PEP-8