Язык программирования Python

Лекция №

Коллекции и функции в PYTHON

Лектор Аксентьев Артем Алексеевич

Что изучим на курсе

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

Что изучим сегодня

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

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++

Сферы программирования

GAMEDEV

Игры, физика,
графика
C++, C#

BUSINESS

Автоматизация
бизнеса
1C, SAP

SCIENCE

Машинное обучение,
моделирование систем
R, Python

Программирование сегодня

- **АЛГОРИТМЫ**

  Умение оптимизировать решение

- **ПРЕДМЕТНАЯ ОБЛАСТЬ**

- **УПРАВЛЕНИЕ**

  Распределение заданий, сил и т.д.

- **ИНСТРУМЕНТЫ**

  Языки программирования, фреймворки, библиотеки

Python 3

CPython

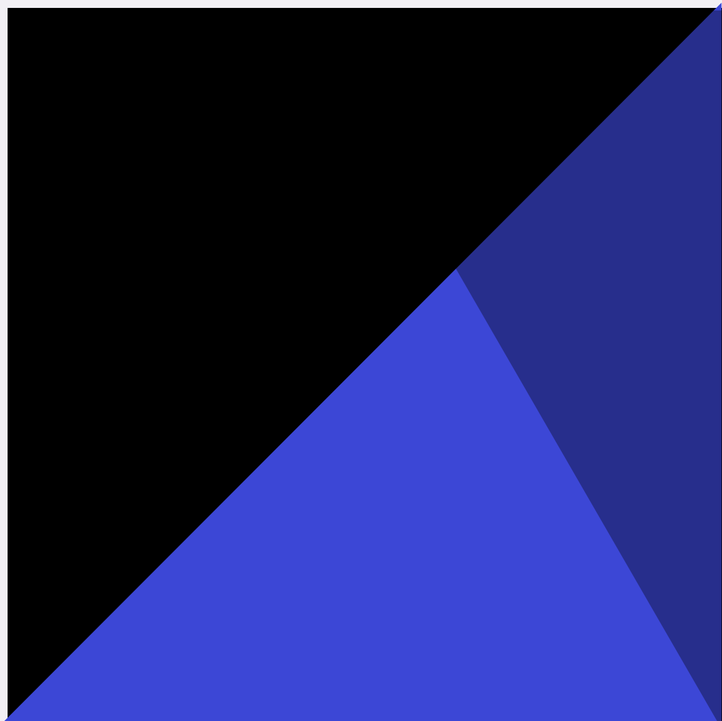
Интерпретатор написанный на Си

Jython

Интерпретатор написанный на Java, можно использовать некоторые фичи стандартной библиотеки Java

PyPy

Интерпретатор написан на Python



КОМПЬЮТЕРНОЕ МЫШЛЕНИЕ

ВХОДНЫЕ ДАННЫЕ

АЛГОРИТМЫ

ВЫХОДНЫЕ ДАННЫЕ

Хранение цифр в памяти компьютера

123

Хранение цифр в памяти компьютера

123

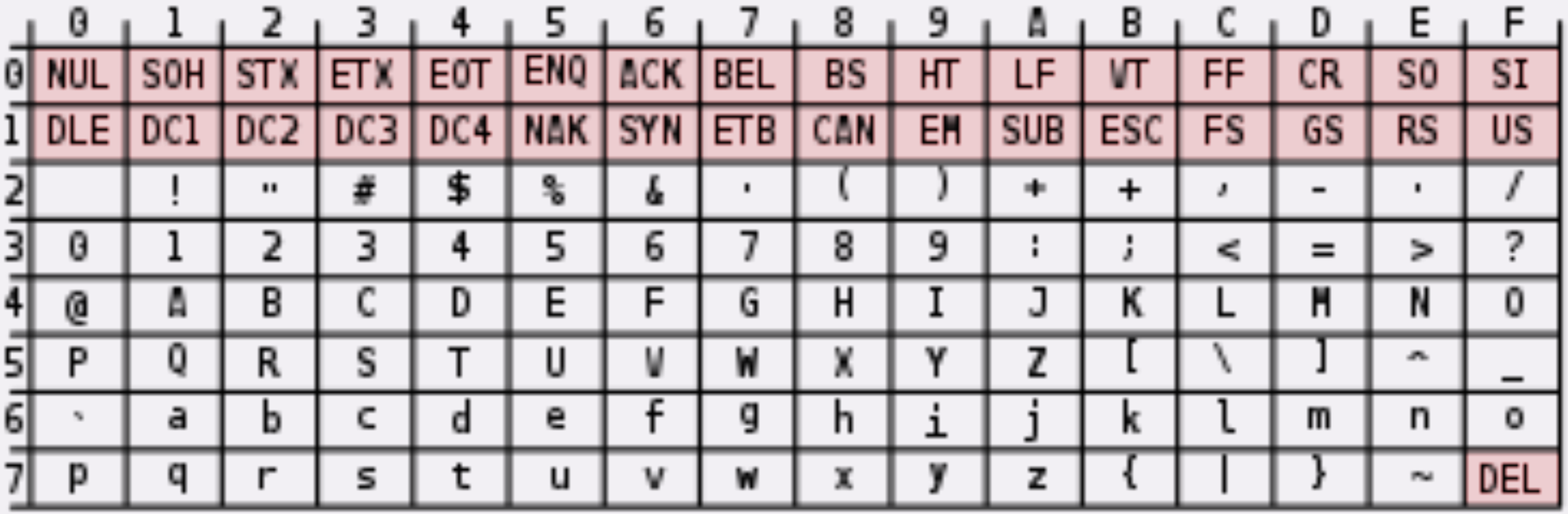
1 * 100 + 2 * 10 + 3 * 1

Хранение цифр в памяти компьютера

101010

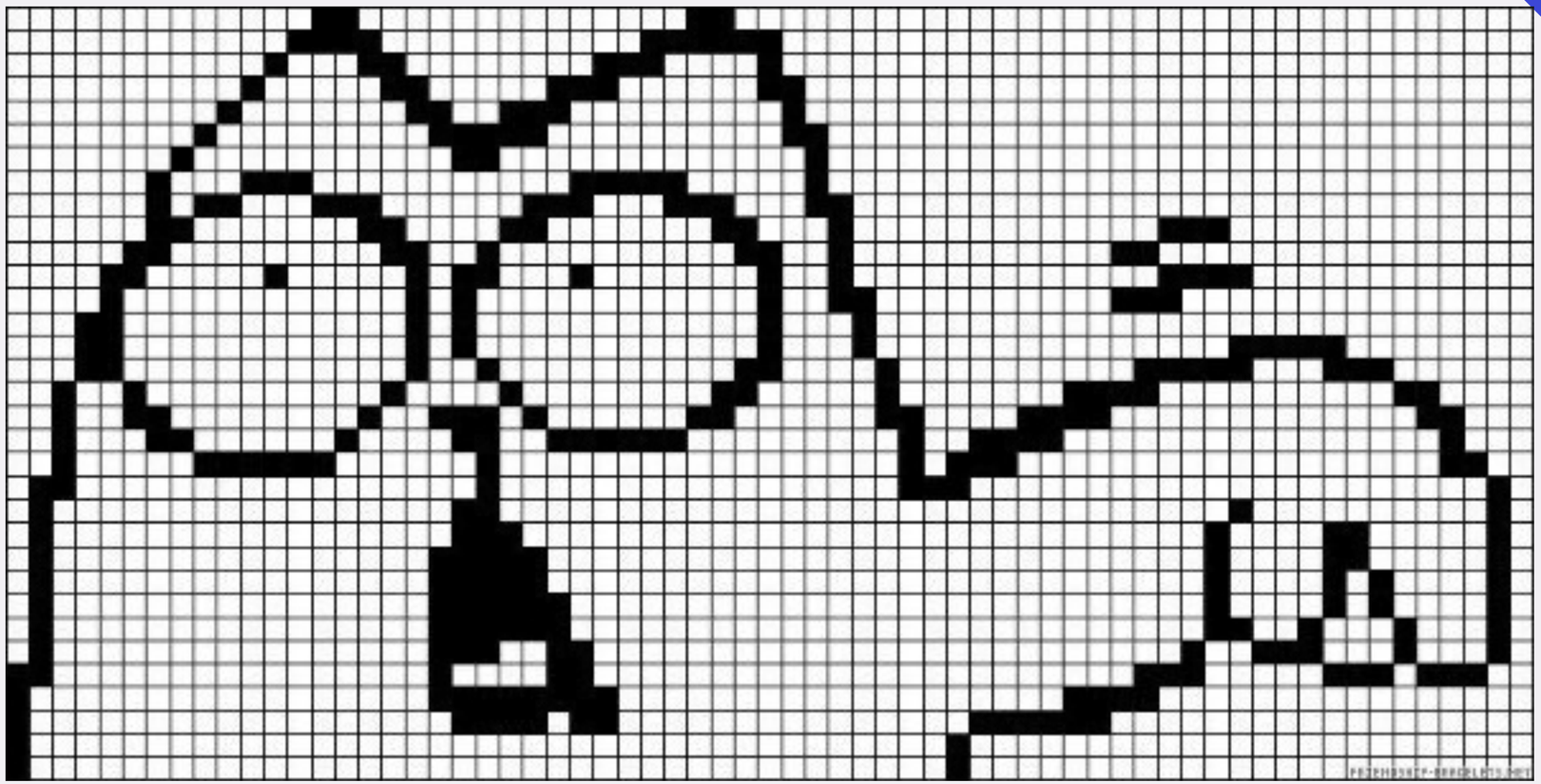
1 * 32 + 0 * 16 + 1 * 8 + 0 * 4 + 1 * 2 + 0 * 1

Хранение букв в памяти компьютера

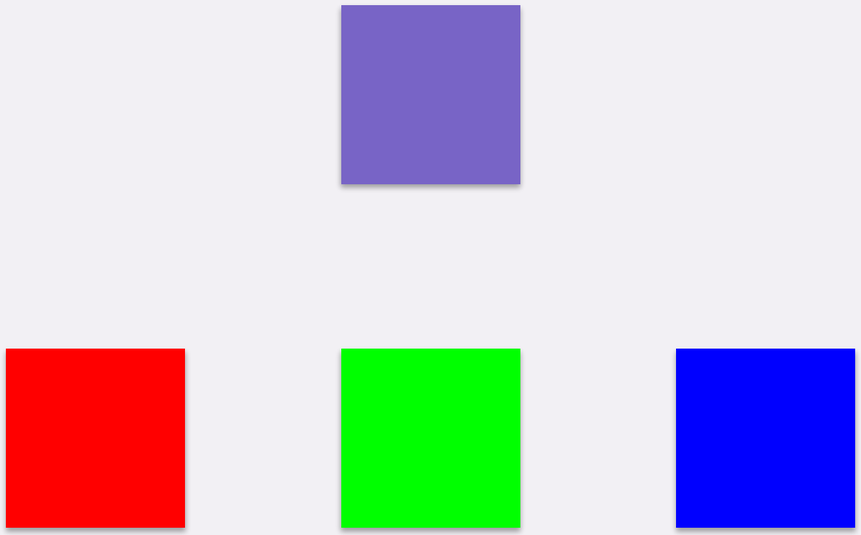


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Хранение изображений в памяти компьютера



Хранение цвета в памяти компьютера



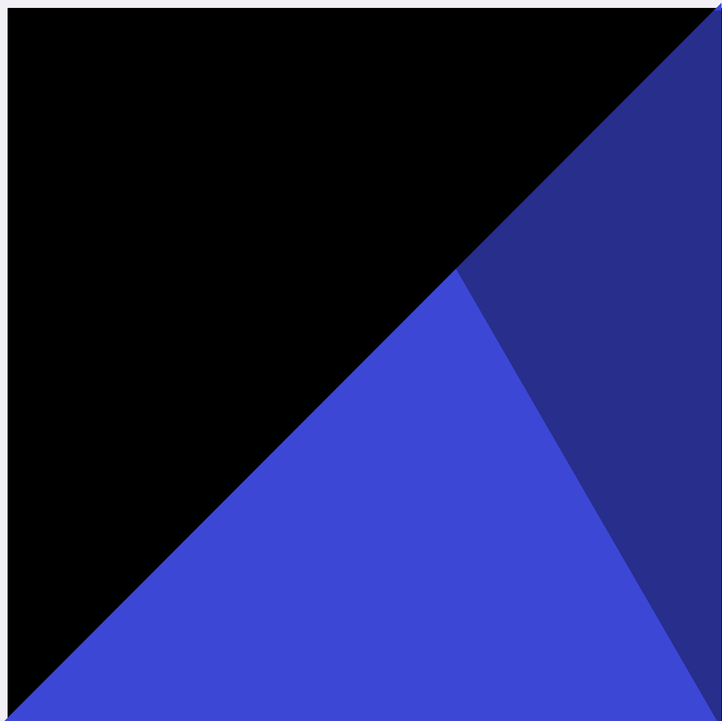
120 100 198

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.

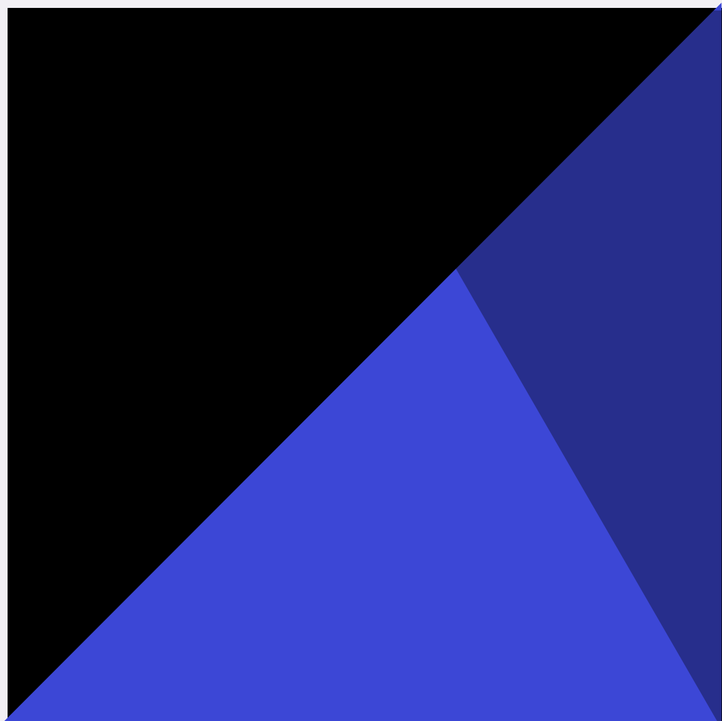
Ваша задача за минимальное время найти пробирку с ядом.



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?

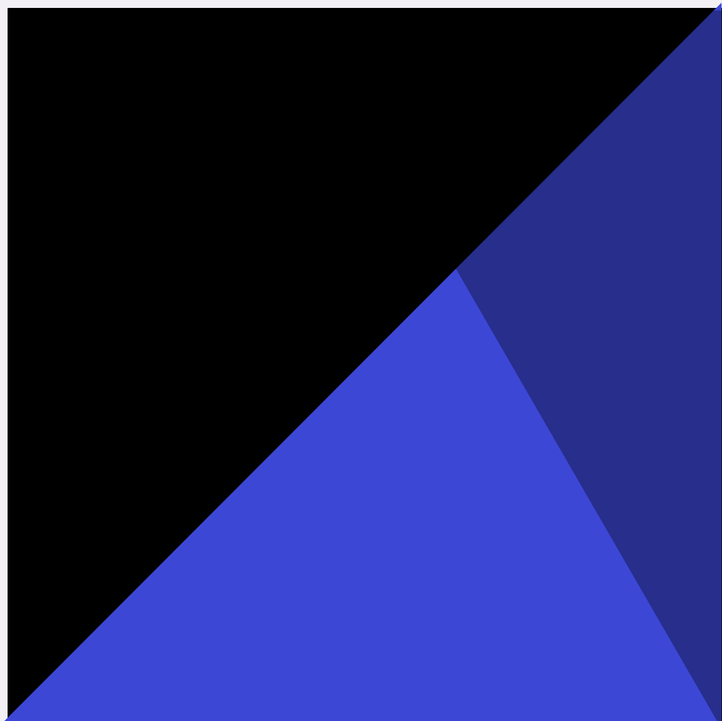


Задача о мышах

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



АЛГОРИТМ

ОПРЕДЕЛЕННОСТЬ

Не должно быть случайных выводов

МАССОВОСТЬ

Работа на разных данных

ПОИСК ЭЛЕМЕНТА

В словаре найти слово "Программирование"

Алгоритмы поиска в отсортированном массиве

ЛИНЕЙНЫЙ

Проверяем каждый элемент

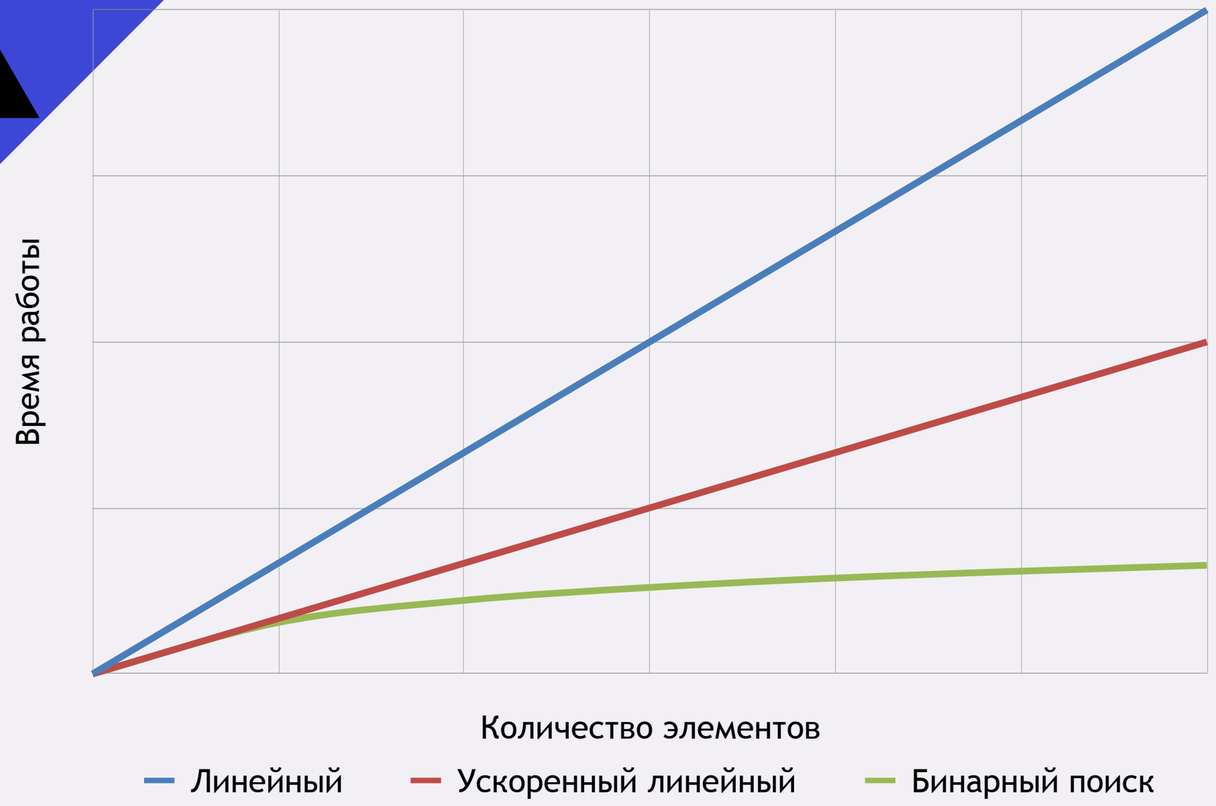
"УСКОРЕННЫЙ"

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

Hello, world

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

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

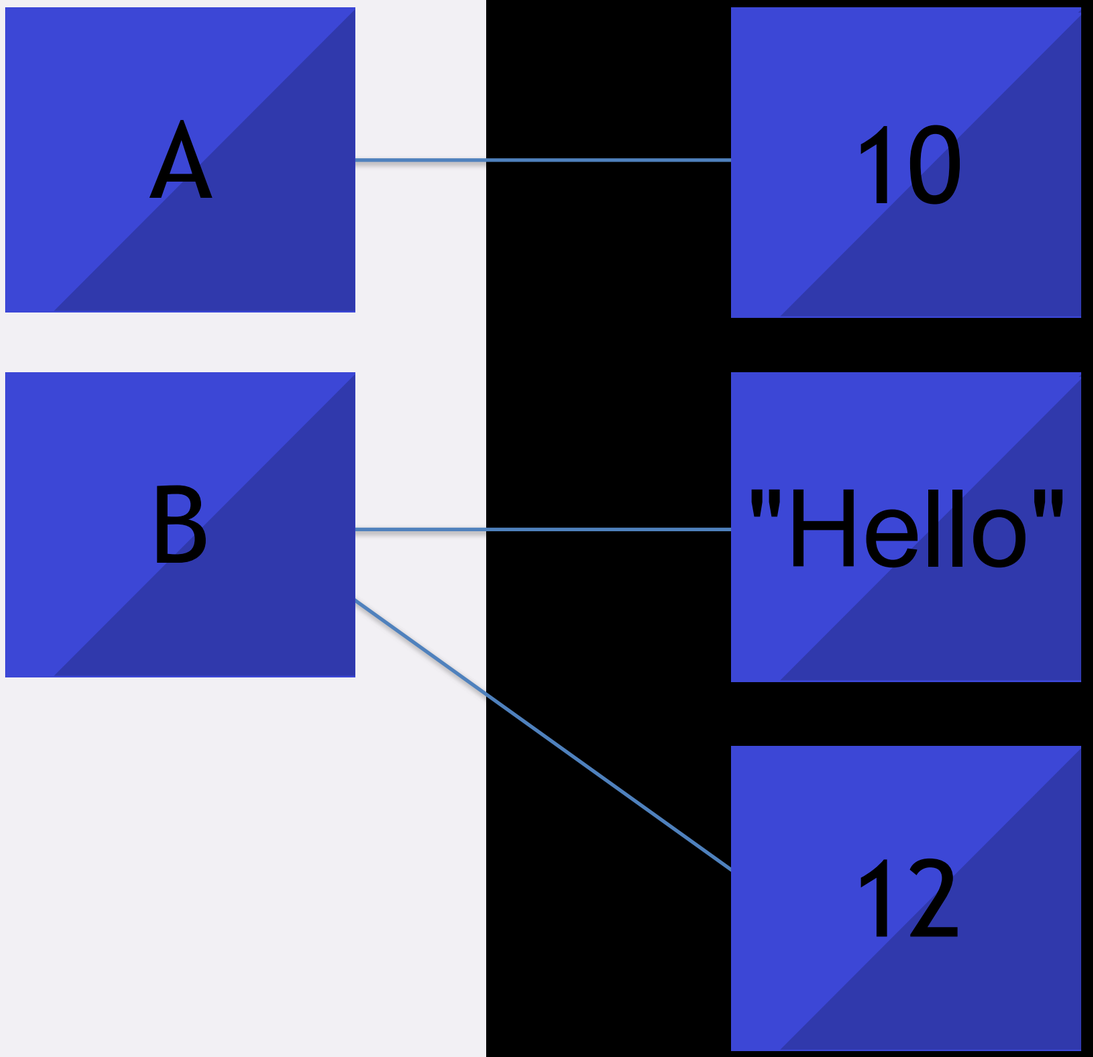
```
>>>name = input("Введите своё имя: ")
Введите своё имя: Иван
>>>print("Привет, ", name)



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

ПЕРЕМЕННЫЕ

ДАННЫЕ



ИЗМЕНЯЕМЫЕ

Списки, множества, словари

НЕИЗМЕНЯЕМЫЕ

Числа, строки, кортежи, логические данные

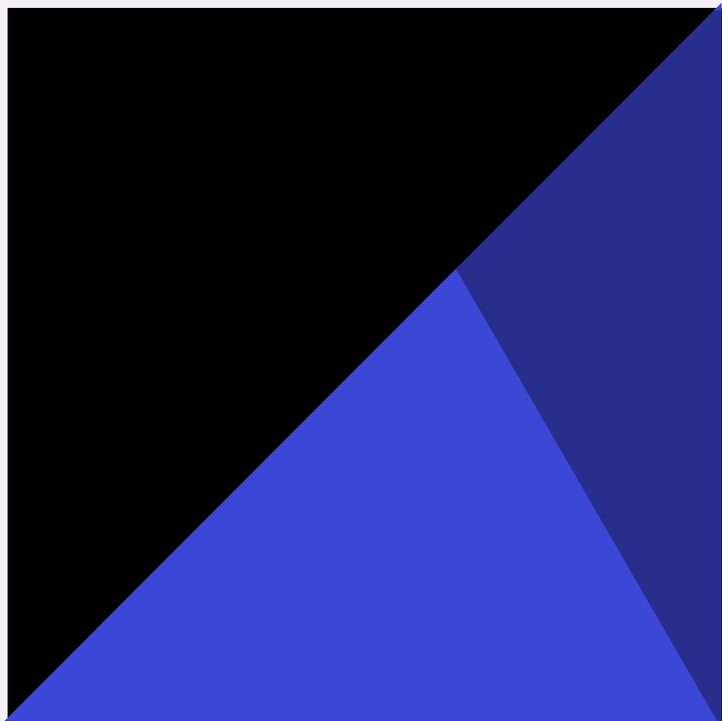
Целые числа

Не имеют ограничений

Не задумывайтесь об этом

Поддерживают стандартные математические операции

x + y, x - y, x / y, x // y, x % y, abs(x), x**y



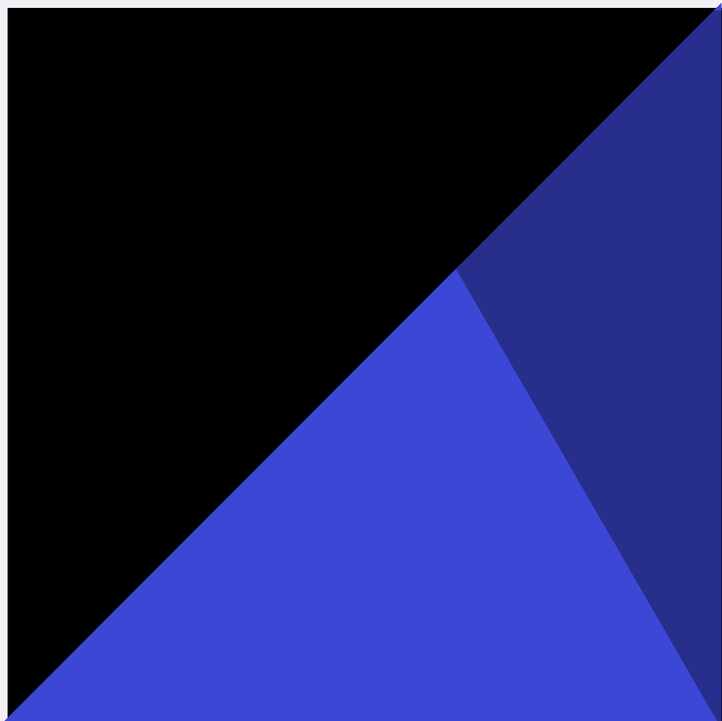
Вещественные числа

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



List

```
>>>a1 = 5
>>>a2 = 6
>>>a3 = 7
>>>a4 = 8
```

List

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

Списки

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

ИМЕНОВАНИЕ ПЕРЕМЕННЫХ

ИМЯ ОТОБРАЖАЕТ СОДЕРЖАНИЕ ПЕРЕМЕННОЙ

a, b, c, q, com, test - не лучший выбор

ВЕРБЛЮЖИЙ РЕГИСТР

longVariable long_variable

ВХОДНЫЕ ДАННЫЕ

Удобно оставлять имена переменных из задачи

CODESTYLE

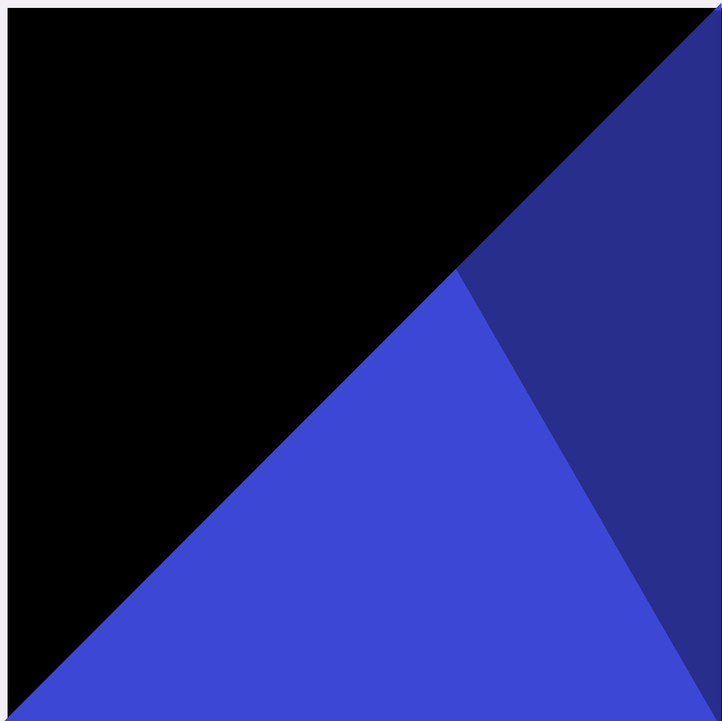
Максимальная длина строки 79 символов

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

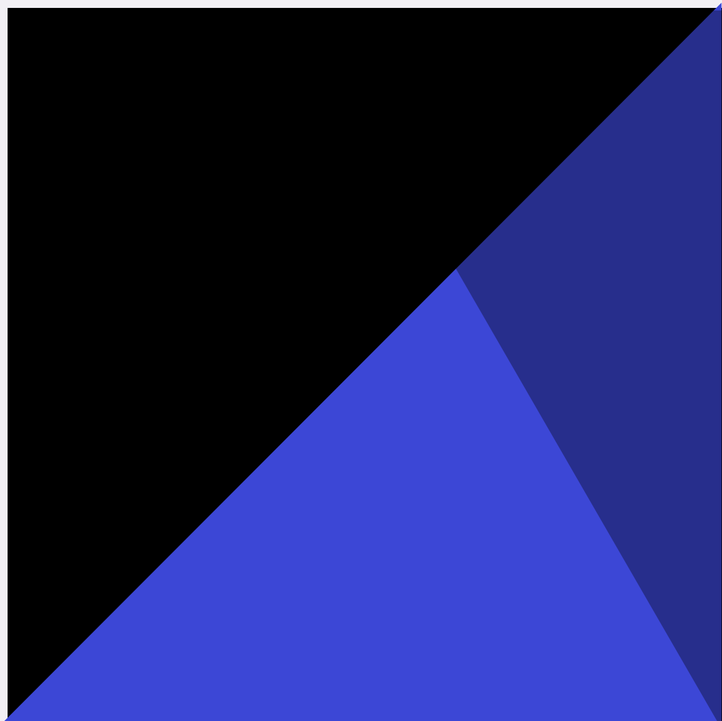
Не противоречат коду

Изменил код, измени и комментарий

Законченные предложения

Кратко, но емко

Комментарии и код по возможности на английском



Conditions

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

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

Conditions

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

Логические ошибки

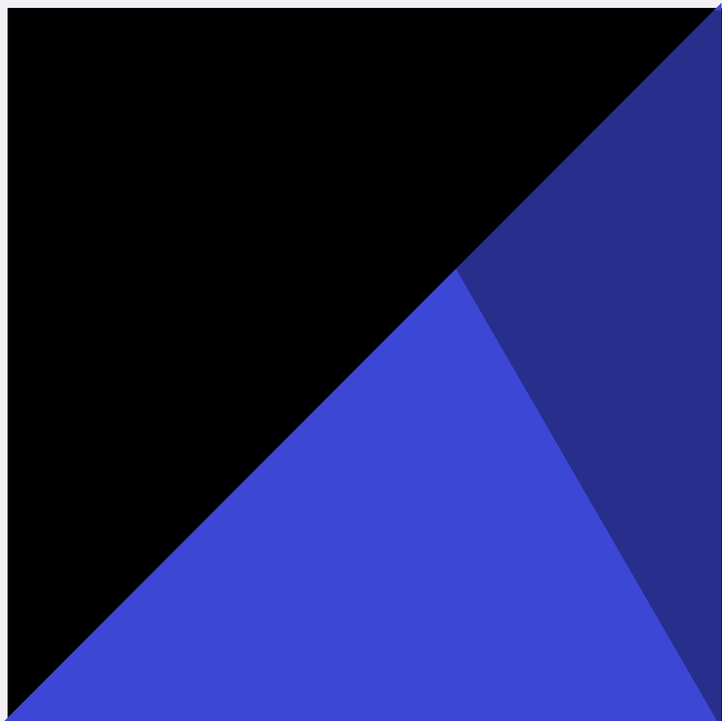
Ошибки в логике работы программы. Проверка данных на некоторых шагах исполнения

Синтаксические ошибки

Ошибки в коде. Интерпретатор покажет их

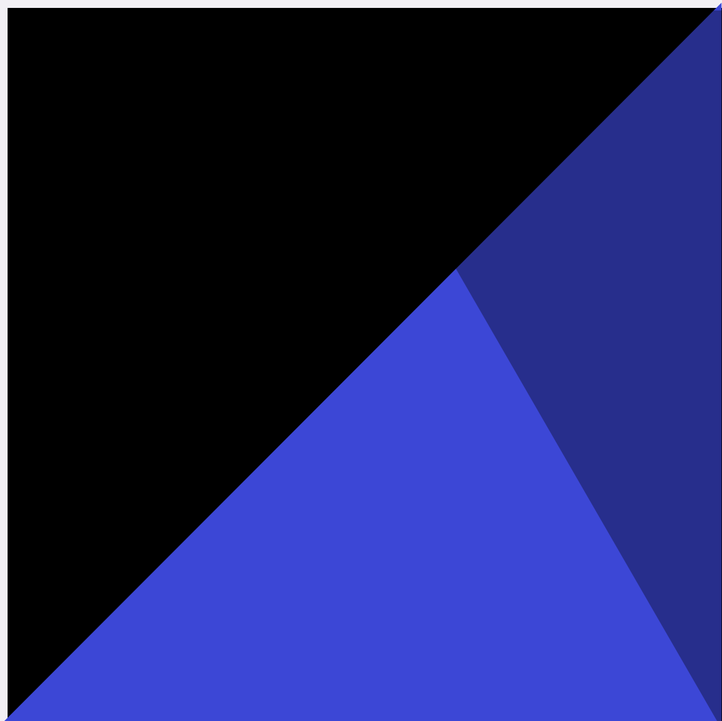
Дзен Python

- 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

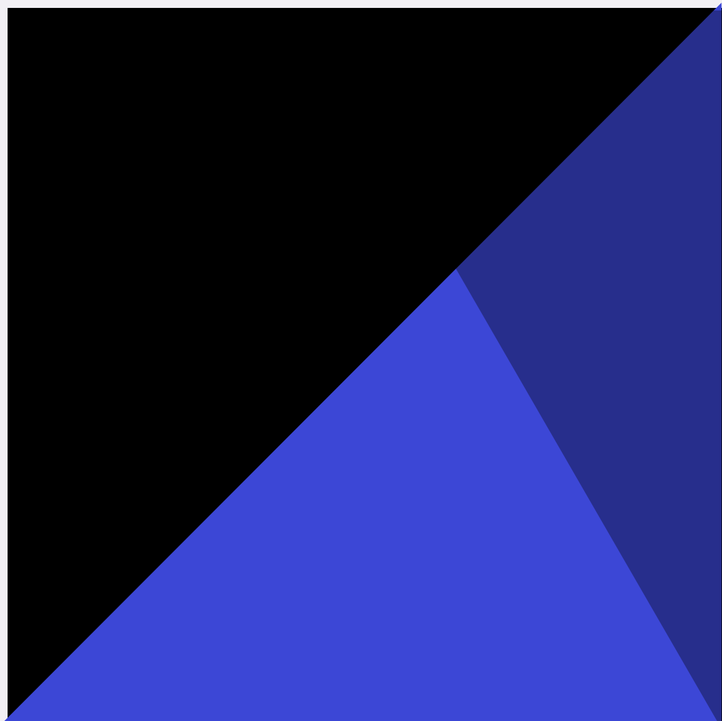
Контроль версий

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

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**
- **Оставляйте сообщение для коммита**
- **Старайтесь, чтобы каждый коммит содержал одно изменение**
- **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

- **Продумайте все возможные варианты**

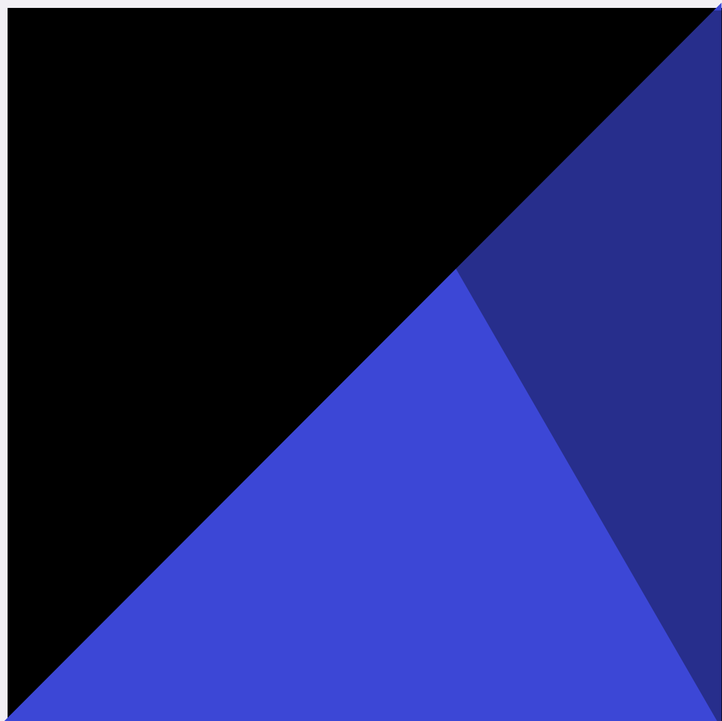
- **Попробуйте оптимизировать**

- **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

Возникла проблема?

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