

# Анализ кода

## Лекция 1 — RegEx и лексический анализ

Лазар В. И.

Школьный курс программирования

17 апреля 2025 г.

# Почему мы изучаем интерпретаторы?

- Python, JavaScript, Lua — все интерпретируемые.
- Быстрый REPL-цикл: пишем → сразу видим результат.
- Даже компиляторы внутри содержат интерпретирующие фазы (AST walk).

# Почему мы изучаем интерпретаторы?

- Python, JavaScript, Lua — все интерпретируемые.
- Быстрый REPL-цикл: пишем → сразу видим результат.
- Даже компиляторы внутри содержат интерпретирующие фазы (AST walk).

- 1 Введение, RegEx, лексический анализ
- 2 Лексер на `ply.lex`
- 3 Парсер на `ply.yacc` и AST
- 4 Выполнение AST, окружения
- 5 Условия, циклы, встроенные функции
- 6 Итоговый REPL, ошибки, перспективы

- Итог — интерпретатор языка с переменными, условиями и циклами.
- Оценивание по ТЗ (до 100 б.).

# Регулярные выражения: экспресс-повторение

Метасимволы	.	любой символ
	*	0 и более
	+	1 и более
	?	0 или 1
	[abc]	a   b   c
	^[abc]	не a,b,c
		альтернатива
	()	группа

Квантификаторы	{n}	ровно n
	{n,}	$\geq n$
	{,m}	$\leq m$
	{n,m}	от n до m

- Лексер сканирует поток символов и группирует их в токены.
- Каждый токен описывается регулярным выражением.

- Лексер сканирует поток символов и группирует их в токены.
- Каждый токен описывается регулярным выражением.

```
rules = [  
    (r"\d+", "NUMBER"),  
    (r"[A-Za-z_][A-Za-z0-9_]*", "IDENT"),  
    (r"\+", "PLUS"), (r "-", "MINUS"),  
    (r"\*", "STAR"), (r"/", "SLASH"),  
    (r"\(", "LPAREN"), (r"\)", "RPAREN"),  
]
```



# Пример токенизации строки

Исходная строка:  $1 + 2 * (3 - 4)$

Токенизированная строка:

[NUMBER(1), PLUS, NUMBER(2), STAR, LPAREN,  
NUMBER(3), MINUS, NUMBER(4), RPAREN]

- Пробелы и комментарии игнорируются.
- Позиции токена: строка, колонка — нужны для сообщений об ошибках.

# Почему PLY?

- Чистый Python, проще ставить и отлаживать
- Поддерживает LALR(1)-грамматики, достаточно для TinyPy.
- Правила задаются обычными функциями.

Текст → **Лексер (PLY)** → токены → **Парсер (PLY)** → AST → **Выполнение.**

# Практическое задание к занятию 1

Реализовать минимальный лексер TinyPy:

- 1 Склонировать репозиторий, открыть `tinypy/lexer.py`.
- 2 Добавить правила для токенов: `NUMBER`, `IDENT`, `+` `-` `*` `/` `(` `)` `{` `}` `=` и ключевых слов `let`, `if`, `else`, `print`, `while`.
- 3 Игнорировать пробелы и комментарии `# ....`
- 4 Обновлять `lineno` и вычислять колонку.
- 5 Запустить `pytest tests/test_lexer_basic.py` — 16 тестов должны пройти.