TinyPy — Техническое задание

Устройство интерпретатора

17 апреля 2025 г.

Содержание

1 Что такое TinyPy

TinyPy — маленький учебный язык, похожий на сильно упрощённый Python. Он умеет:

- считать числовые выражения;
- хранить числа в переменных;
- выполнять условия `if`/`else` и циклы `while`;
- выводить данные через `print`.

Финальный результат — программа на Python, запускаемая так:

```
python -m tinypy              # интерактивный режим
python -m tinypy prog.tny     # исполнить файл TinyPy
```

2 Структура проекта

```
tinypy/
 --- lexer.py        # превращает текст в токены
 --- parser.py       # собирает токены в дерево (AST)
 --- ast_nodes.py    # классы узлов AST
 --- interpreter.py  # выполняет дерево
 --- repl.py         # интерактивный >>-цикл
 --- errors.py       # классы ошибок (Lexer-, Syntax-, Runtime-)

tests/               # автотесты преподавателя
```

3 Токены, которые обязан узнавать лексер

Категория	Вид	Пример
Число	цифры (можно с точкой)	`12`, `3.14`
Идентификатор	буква/_ + буквы/цифры/_	`x`, `total_sum`
Ключевые слова	`let`, `if`, `else`, `while`, `true`, `false`, `print`	
Односимвольные знаки	`+ - * / ( ) { } ;`	
Двухсимвольные знаки	`== != < <= > >= =`	
Комментарии	# до конца строки	`# это комментарий`

Пробелы и табы игнорируются.

4 Синтаксис TinyPy

4.1 Программа

Набор команд, разделённых переводом строки или символом `;`. Пустые строки разрешены.

4.2 Переменные

```
let radius = 10
let area   = 3.14 * radius * radius
```

- Каждое объявление начинается со слова `let`.
- Имя можно объявить только один раз внутри одного блока.

4.3 Выражения

- Числа, имена переменных, `true`/`false`.
- Операции (от старшего приоритета к младшему):

$$** \quad \rightarrow \quad */ \quad \rightarrow \quad +- \quad \rightarrow \quad ==! =<<=>>=$$

- Скобки `( ... )` меняют порядок вычислений.

4.4 Команда-выражение

Если написать выражение отдельно, результат выводится автоматически:

```
1 + 2   # печатает 3
```

4.5 Функция `print`

```
print(1 + 2)
print(radius)
```

4.6 Условие

```
if (x > 0) {
    print(x)
} else {
    print(0)
}
```

4.7 Цикл

```
while (n > 0) {
    print(n)
    n = n - 1
}
```

4.8 Блоки

Код внутри `{ ... }` создаёт новый «вложенный» набор переменных.

5 Ошибки

Ситуация	Бросить исключение
Неизвестный символ	`LexerError`
Неправильная последовательность токенов	`SyntaxError_`
Деление на 0, неизвестная переменная	`RuntimeError_`

Сообщение ошибки должно содержать хотя бы номер строки.

6 Оценивание

Раздел	Баллы
Лексер (лекции 1–2)	25
Парсер + AST (лекция 3)	25
Интерпретатор (лекции 4–5)	40
REPL + запуск из файла (лекция 6)	10
Итого	**100**

Большую часть баллов выставляют автотесты; остальное — проверка кода преподавателем.

7 Инструменты

- Python 3.11, библиотека `PLY`.
- Тесты — `pytest`, стиль — `flake8`, проверка опечаток — `codespell`.
- GitHub Actions уже настроен: при каждом `push` запускаются тесты, линтер и spell-check.

8 Формат сдачи

Создаётся отдельная личная ветка в репозитории. Все изменения производятся исключительно в своей ветке. Для сдачи определённого этапа создаётся Pull Request.