«Теория автоматов и формальных языков»

тема: Создание лексического анализатора

Санкт-Петербург, 2025

**Задание**

1. Cоставить таблицу токенов. Таблица может не быть исчерпывающей.

2. Для каждого типа токенов придумать и описать паттерн, которому соответствуют значения такого токена. Для этих целей можно использовать регулярные выражения или другой способ, кажущийся вам более подходящим.

3. Написать программу, которая будет получать на вход строку, а на выход выдавать последовательность токенов (с указанием как имени токена, так и его значения).

Важно! На этапе лексического анализа не проверяется корректность поданной на вход строки, только разбиение ее на токены.

**Формирование токенов**

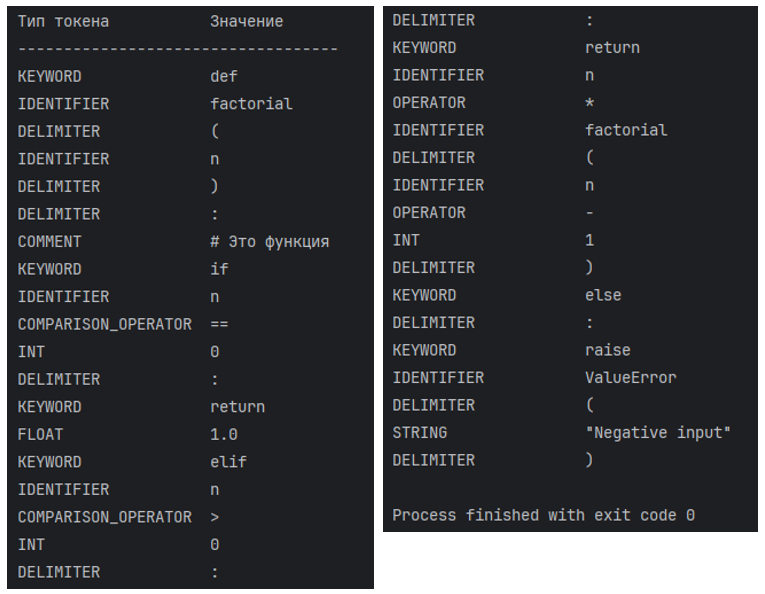
|  |  |  |
| --- | --- | --- |
| **Имя токена** | **Значения токена** | **Шаблон токена** |
| KEYWORD | if, else, elif, for, while, def, return, and, or, not, in, is, break, continue, as, try, raise, except, True, False, import, from | \b(if|else|elif|for|…|False)\b |
| DELIMITER | ()[]{}:, | (\(|\)|\[|\]|\{|\}|\,|\:) |
| OPERATOR | = + - \* / \*\* // % | (?:=|\+|-|\\*|\/|\\*\\*|\/\/|%) |
| COMPARISON OPERATOR | == != > < >= <= | (?:<=|>=|==|!=|>|<) |
| LOGICAL OPERATOR | & | ^ | (?:&|\||\^) |
| IDENTIFIER | x, i, count, Tmatrix\_1 | \b[a-zA-Z\_][a-zA-Z0-9\_]\*\b |
| INT | 10, -10 | \-?\d+\b |
| FLOAT | 3.14, -0.5 | \-?\d+\.\d+ |
| STRING | "hello", 'world' | (?:'[^']\*'|\"[^\"]\*\") |
| SPACE |  | [\ ] |
| COMMENT | # qwerty """qwerty""" | (?:\#.\*$|\"\"\"[^\"\"\"]\*\"\"\") |

**Результаты работы программы**

Далее будут приведены примеры проанализированных строк программ и таблица соответствия значения токена его типу.

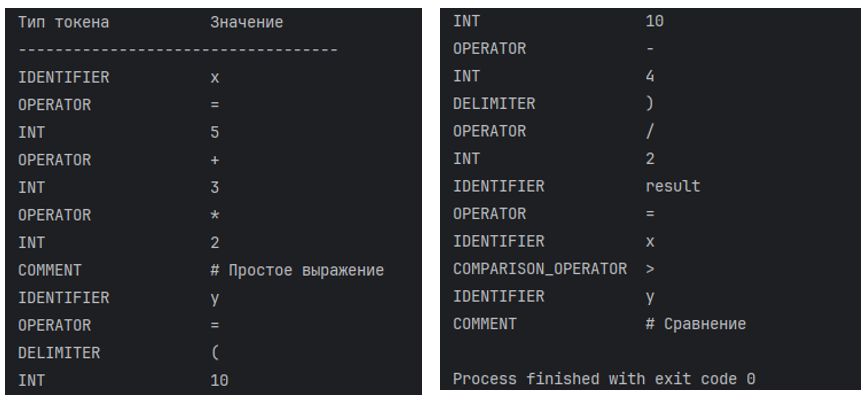
Вычисление факториала:

|  |
| --- |
| def factorial(n): # Это функция  if n == 0:  return 1.0  elif n > 0:  return n \* factorial(n - 1)  else:  raise ValueError("Negative input") |



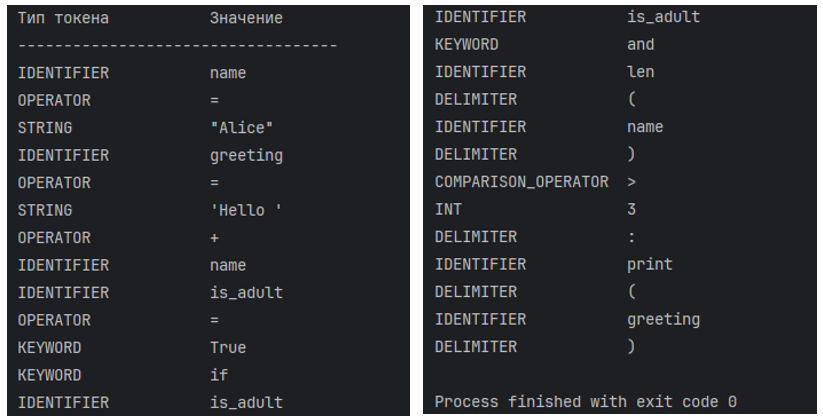
Простые арифметические операции:

|  |
| --- |
| x = 5 + 3 \* 2 # Простое выражение y = (10 - 4) / 2 result = x > y # Сравнение |



Работа со строками и логические операторы:

|  |
| --- |
| name = "Alice" greeting = 'Hello ' + name is\_adult = True if is\_adult and len(name) > 3:  print(greeting) |



**Вывод**

В ходе выполнения работы был разработан лексический анализатор (лексер) для языка Python, который разбивает исходный код на токены — базовые элементы языка (ключевые слова, идентификаторы, операторы, числа, строки и др.).

**Листинг программы**

|  |
| --- |
| import re  # Определение токенов в порядке приоритета TOKENS = [  ('SPACE', r'[\s]+'), # Пробельные символы (игнорируются)  ('COMMENT', r'\#.\*'), # Комментарии (игнорируются)  ('KEYWORD', r'\b(if|else|elif|for|while|def|return|and|or|not|in|is|break|continue|as|try|raise|except|True|False|import|from)\b'),  ('COMPARISON\_OPERATOR', r'<=|>=|==|!=|>|<'),  ('OPERATOR', r'=|\+|-|\\*|\/|\\*\\*|\/\/|%'),  ('LOGICAL\_OPERATOR', r'&|\||\^'),  ('DELIMITER', r'\(|\)|\[|\]|\{|\}|\,|\:'),  ('FLOAT', r'-?\d+\.\d+'),  ('INT', r'-?\d+\b'),  ('STRING', r'''('[^']\*'|"[^"]\*")'''),  ('IDENTIFIER', r'\b[a-zA-Z\_][a-zA-Z0-9\_]\*\b'), ]  def tokenize(code):  tokens = [] # Список для хранения найденных токенов  while code:  for token\_type, pattern in TOKENS:  regex = re.compile(pattern) # Компилируем regexp для текущего типа токена  match = regex.match(code) # Пытаемся найти соответствие регулярному выражению в начале строки  if match:  value = match.group(0)  if token\_type not in ['SPACE']: # Пропускаем пробелы  tokens.append((token\_type, value))  code = code[len(value):] # Укорачиваем анализируемую строку, удаляя обработанную часть  break  else: # Если ни один токен не совпал  # Если ни один шаблон не совпал, значит встретился неизвестный символ  # Выводим ошибку с указанием проблемного символа и его позиции  raise SyntaxError(f"Неизвестный символ: '{code[0]}' в позиции {len(code)}")  return tokens  # Пример использования python\_code = ''' name = "Alice" greeting = 'Hello ' + name is\_adult = True if is\_adult and len(name) > 3:  print(greeting) '''  try:  tokens = tokenize(python\_code)  print("{:<20} {:<15}".format("Тип токена", "Значение")) # Вывод в табличном формате  print("-" \* 35)  for token in tokens:  print("{:<20} {:<15}".format(token[0], token[1])) except SyntaxError as e: print(f"Ошибка: {e}") |