Лексический синтаксис

1. <ident> Идентификатор — непустая последовательность букв латинского алфавита в любом регистре, цифр и символа нижнего подчеркивания (_), начинающаяся на букву латинского алфавита в нижнем регистре, не являющаяся ключевым словом.

   * Корректные идентификаторы: x, list, listNat_123.
   * Некорректные идентификаторы: Abc, 123, _List.

2. <num> Число: натуральное или ноль в десятичной системе счисления, не может содержать лидирующие нули.

   * Корректные числа: 123, 0.
   * Некорректные числа: -1, 007, 89A.

3. <keyword> Ключевые слова не могут быть идентификаторами. Конкретные ключевые слова вы выбираете сами.

4. <operator> Операторы языка:
   * сложение +,
   * умножение *,
   * деление /,
   * вычитание -,
   * возведение в степень **,
   * конъюнкция &&,
   * дизъюнкция ||,
   * логическое отрицание --,
   * операторы сравнения: <, <=, ==, /=, >, >=

5. Пробелы не являются значимыми, но не могут встречаться внутри одной лексемы.

Конкретный синтаксис

1. Ключевые слова:

   def - инструкция для определения функции
   pass - пустая инструкция (ничего не делает)
   return - инструкция для возвращения значения из функции
   If - инструкция для определения ветки с условным выраженем
   else - инструкция для определения опциональной ветки условного выражения
   while - инструкция для определения цикла с предусловием

2. Определение функции содержит ее сигнатуру и тело. Сигнатура функции содержит ее название (идентификатор) и список аргументов (может быть пустым). Тело — последовательность инструкций (может быть пустой).

   ```
   def <ident>([<ident>, [<ident>, [...]]]):
     <body>
   ```

   Тело функции указывается с отступом в 4 пробела.

   Примеры:

   - ```
     def sum(a, b):
       return a + b
     ```

   - ```
     def fib(n):
       if n == 1 || n == 2:
         return 1
       if n == 0:
         return 0
       f1 = 0
       f2 = 1
       i = 1
       while i <= n:
         f1 = f2 + f1
         f2 = f1 - f2
         i = i + 1
       return f1
     ```

3. Программа — непустая последовательность определений функций.

   ```
   def <ident>([<ident>, [<ident>, [...]]]):
     <body>

   [
   def <ident>([<ident>, [<ident>, [...]]]):
     <body>
   [
   …
   ]
   ]
   ```

4. Инструкции:

- Присвоение значения арифметического выражения переменной. Переменная может быть произвольным идентификатором.

  Примеры:
  a = 1 + 2
  a = b * c

- Возвращение значения из функции.

  Примеры:
  return a
  return 1 + 2

- Условное выражение с опциональной веткой else. Условием является арифметическое выражение. В ветках — произвольные последовательности инструкций (могут быть пустыми).

  Тело условного выражения указывается с отступом в 4 пробела.

  Примеры:
  ```
  if n == 1 || n == 2:
     return 1
  else:
     return 2
  ```

  Для обозначения пустой ветки используется инструкция *pass*

  ```
  if i == 4:
     pass
  else:
     return sum
  ```

- Цикл с предусловием. Условием является арифметическое выражение. Тело цикла — произвольная последовательность инструкций (может быть пустой).

  Тело цикла указывается с отступом в 4 пробела.

  Пример:

  ```
  i = 0
  while i <= n:
     f1 = f2 + f1
     f2 = f1 - f2
  ```

```
    i = i + 1
```

Для обозначения пустой ветки используется инструкция *pass*

```
i = 0
while i <= n:
    pass
```