Проект SPython

Мовчан Е.В.

Южный федеральный университет,
Институт математики, механики и компьютерных наук

emovchan@sfedu.ru

Доклад на пятой Всероссийской научно-методической конференции
«Использование системы программирования PascalABC.NET
в обучении программированию»
(Ростов-на-Дону, 28-29 марта 2025 г.)

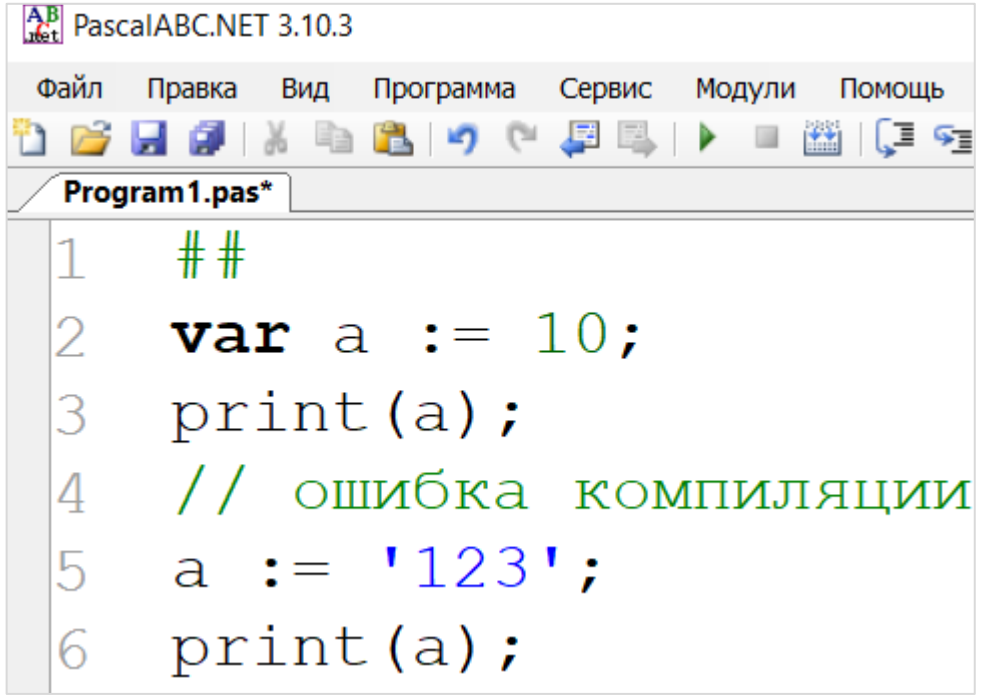
Что такое SPython?

Основные понятия

Виды типизации

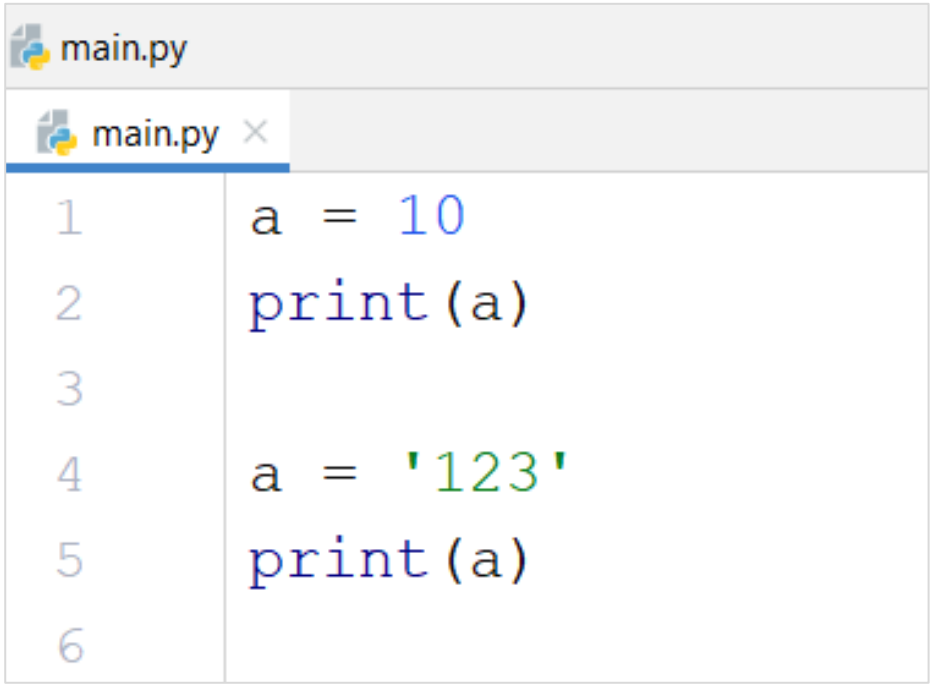
Статическая типизация:

у переменной неизменяемый тип, известный на этапе компиляции



Динамическая типизация:

тип переменной может меняться в процессе выполнения программы



Плюсы статической типизации

- Быстрая и точная диагностика ошибок
- Предсказуемость кода
- Поддержка различных инструментов разработки
- Автоматическое завершение кода и подсказки
- Лучшее сопровождение и поддержка кода
- Повышенная надежность программы
- Улучшение производительности

Компилятор и интерпретатор

Компилятор — программа, которая преобразует исходный код программы, написанный на языке программирования, в машинный код или промежуточный код, который затем выполняется компьютером.

Интерпретатор — программа, которая выполняет исходный код построчно, переводя его в машинный код и сразу же выполняя, без предварительного компилирования в отдельный файл.

Плюсы компиляторов

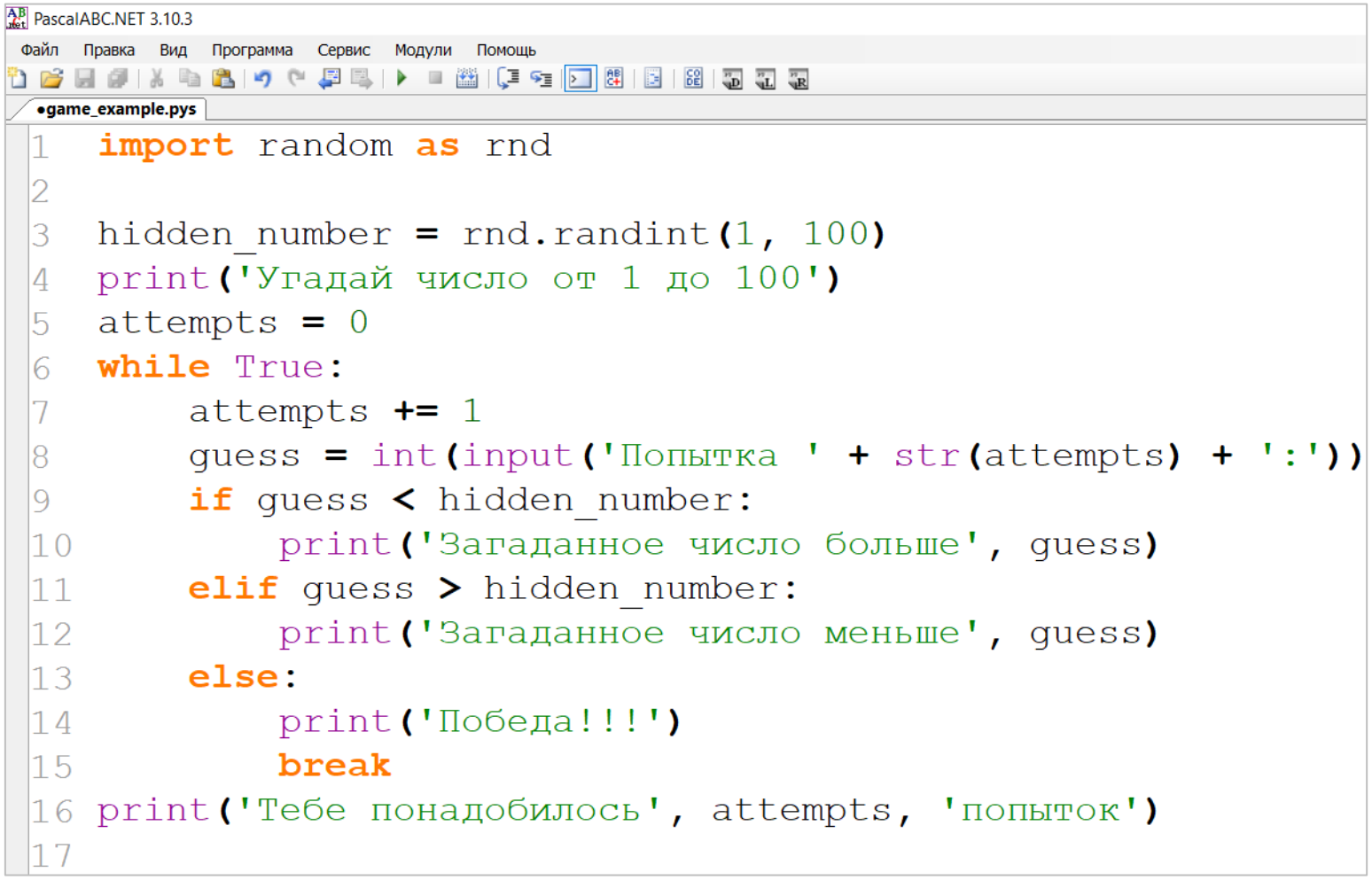
- Высокая производительность кода
- Раннее обнаружение ошибок
- Реализация огромного спектра оптимизаций
- Более строгие инструменты разработки
- Поддержка статического анализа кода
- Гарантии корректности скомпилированного кода

Система плагинов PascalABC.NET

Плагины на PascalABC.NET

- Языковые пакеты реализуются в инфраструктуре PascalABC.NET как плагины.
- Плагины можно дополнительно установить поверх уже загруженной версии PascalABC.NET с помощью инсталляторов.
- Различные языки могут взаимодействовать друг с другом через программные модули.

Пример плагина – SPython

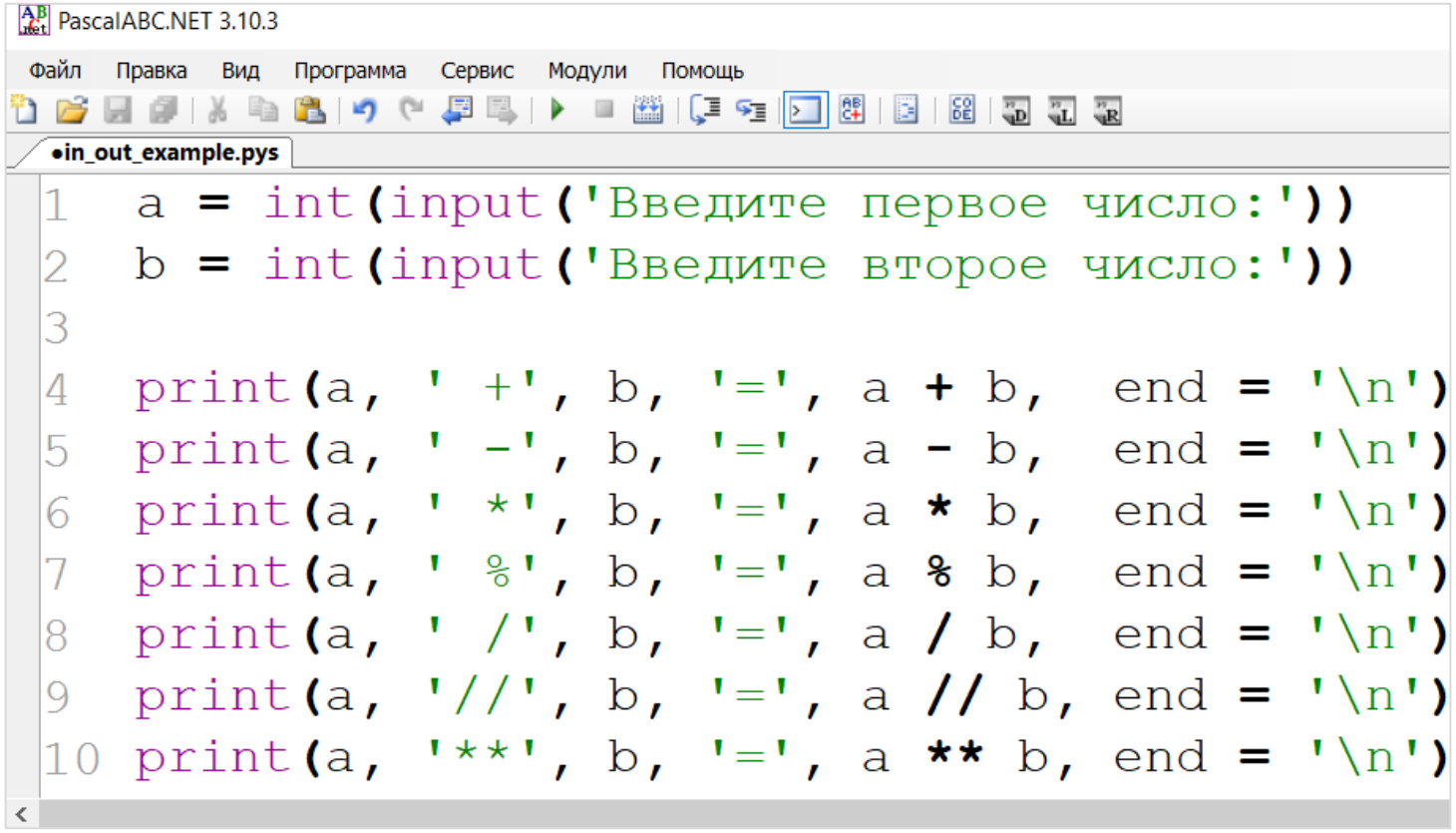


Отличия SPython от Python

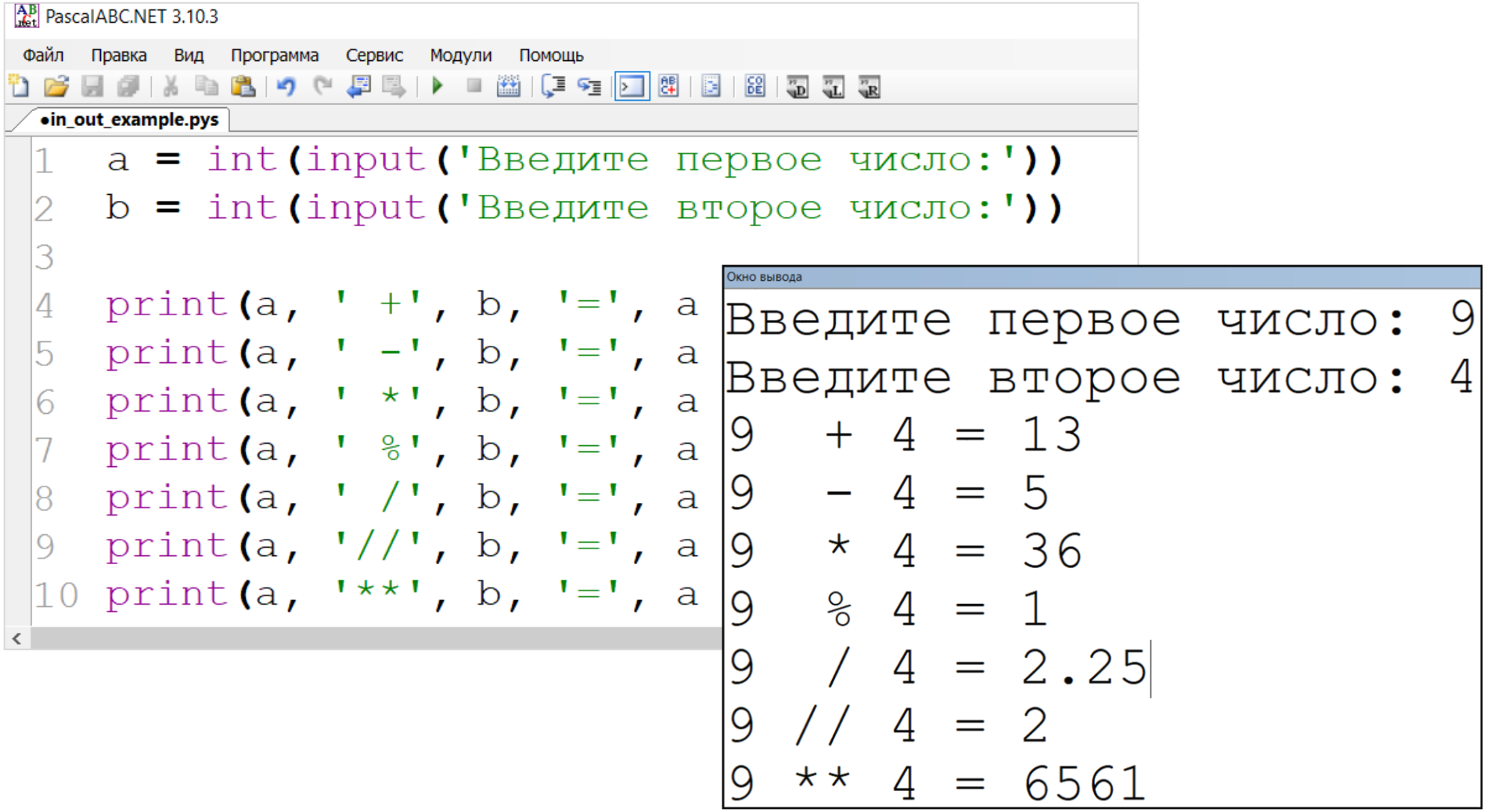
- Статическая типизация
- В SPython принято расширение файлов .pys
- В функциях обязательно указывать типы параметров и тип возвращаемого значения (если оно есть)
- Два типа целых чисел: int и bigint
- Взаимодействие с PascalABC.NET и другими языковыми пакетами
- Не реализовано ООП и часть стандартных функций и конструкций

Примеры программ на SPython

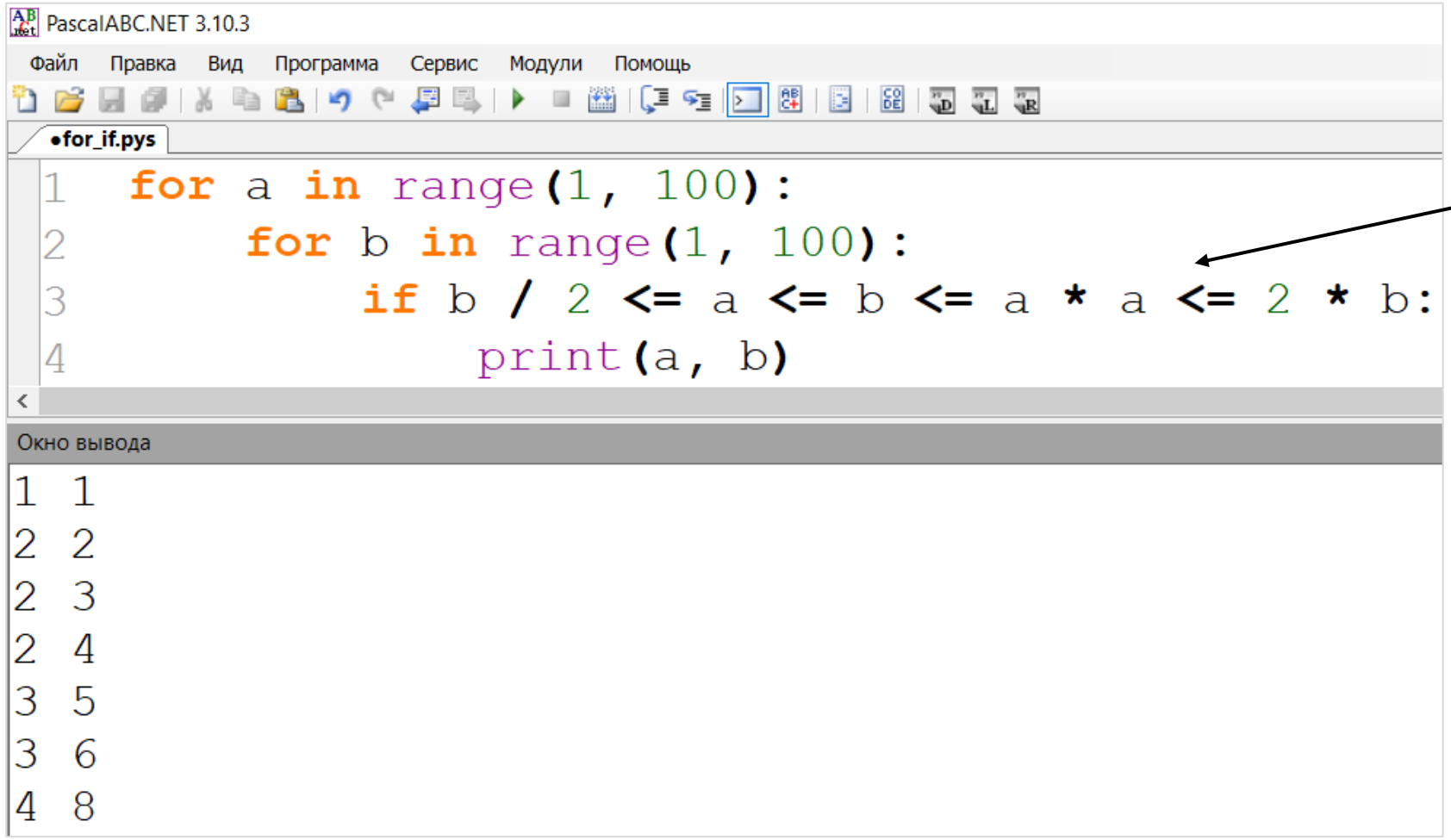
Ввод и вывод


PascalABC.NET 3.10.3
Файл Правка Вид Программа Сервис Модули Помощь
•in_out_example.pys
1 a = int(input('Введите первое число:'))
2 b = int(input('Введите второе число:'))
3
4 print(a, ' +', b, '=', a + b, end = '\n')
5 print(a, ' -', b, '=', a - b, end = '\n')
6 print(a, ' *', b, '=', a * b, end = '\n')
7 print(a, ' %', b, '=', a % b, end = '\n')
8 print(a, ' /', b, '=', a / b, end = '\n')
9 print(a, '//', b, '=', a // b, end = '\n')
10 print(a, '**', b, '=', a ** b, end = '\n')

Ввод и вывод


PascalABC.NET 3.10.3
Файл Правка Вид Программа Сервис Модули Помощь
•in_out_example.pys
1 a = int(input('Введите первое число:'))
2 b = int(input('Введите второе число:'))
3
4 print(a, ' +', b, '=', a
5 print(a, ' -', b, '=', a
6 print(a, ' *', b, '=', a
7 print(a, ' %', b, '=', a
8 print(a, ' /', b, '=', a
9 print(a, '//', b, '=', a
10 print(a, '**', b, '=', a
Окно вывода
Введите первое число: 9
Введите второе число: 4
9 + 4 = 13
9 - 4 = 5
9 * 4 = 36
9 % 4 = 1
9 / 4 = 2.25
9 // 4 = 2
9 ** 4 = 6561

Циклы и условные операторы



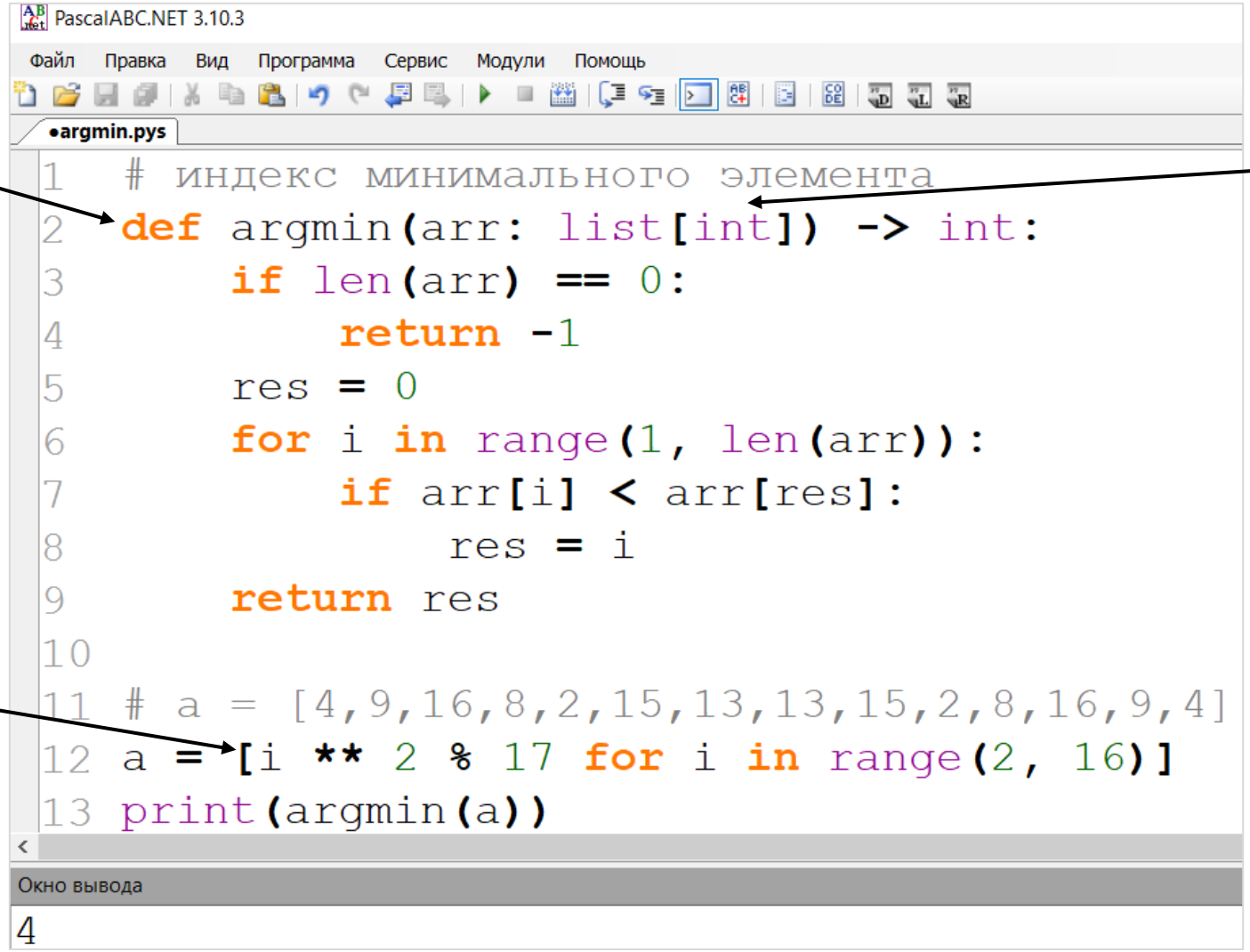
Цепочечные условия

Поиск индекса минимального элемента в списке

Объявление функции

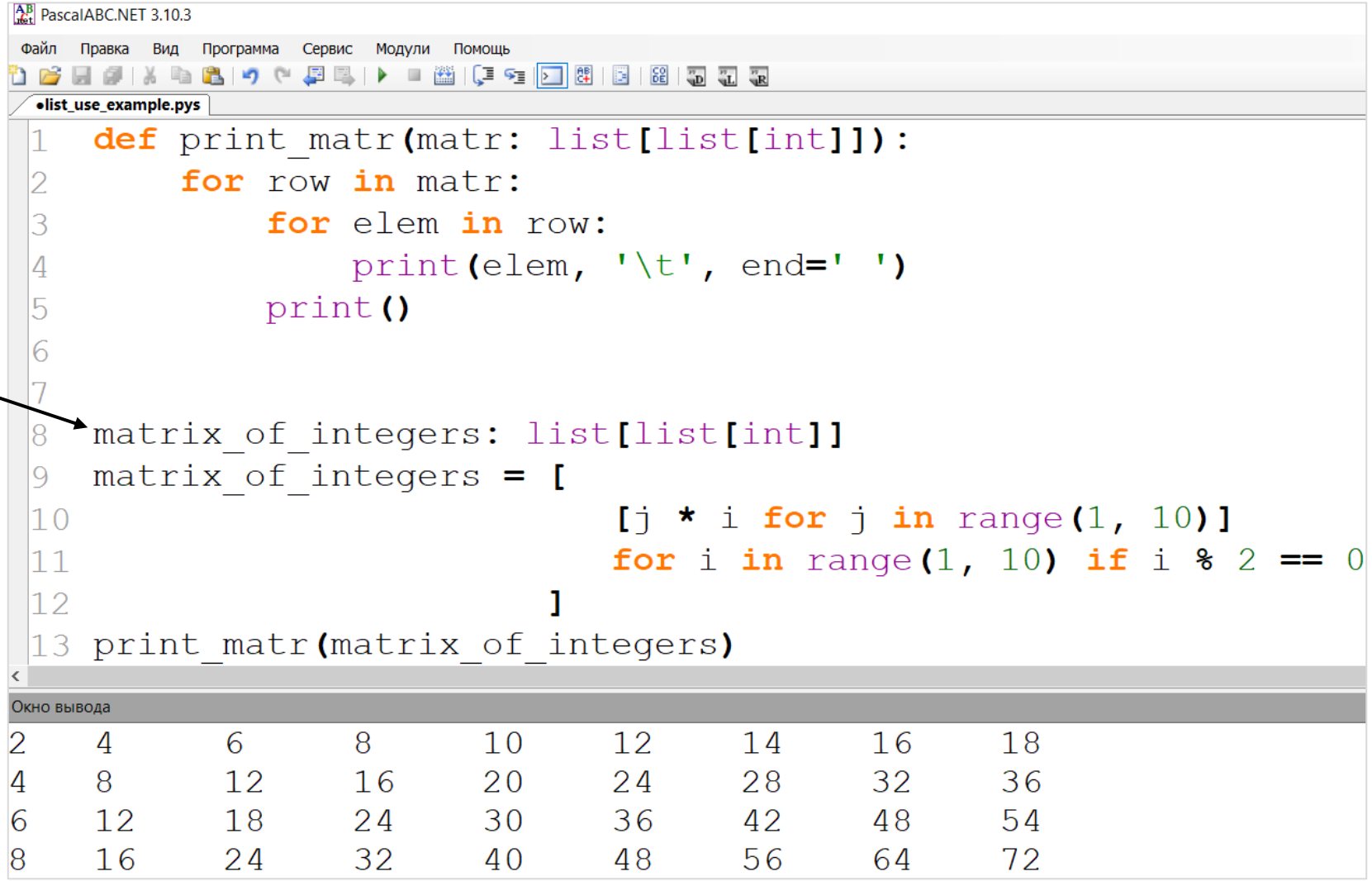
Тип списка

Генератор списка



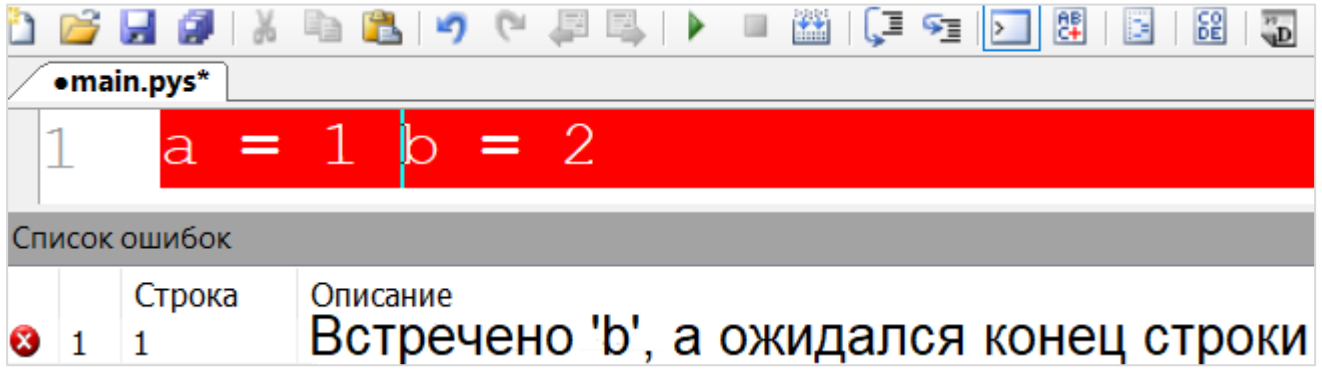
Работа с двумерными списками

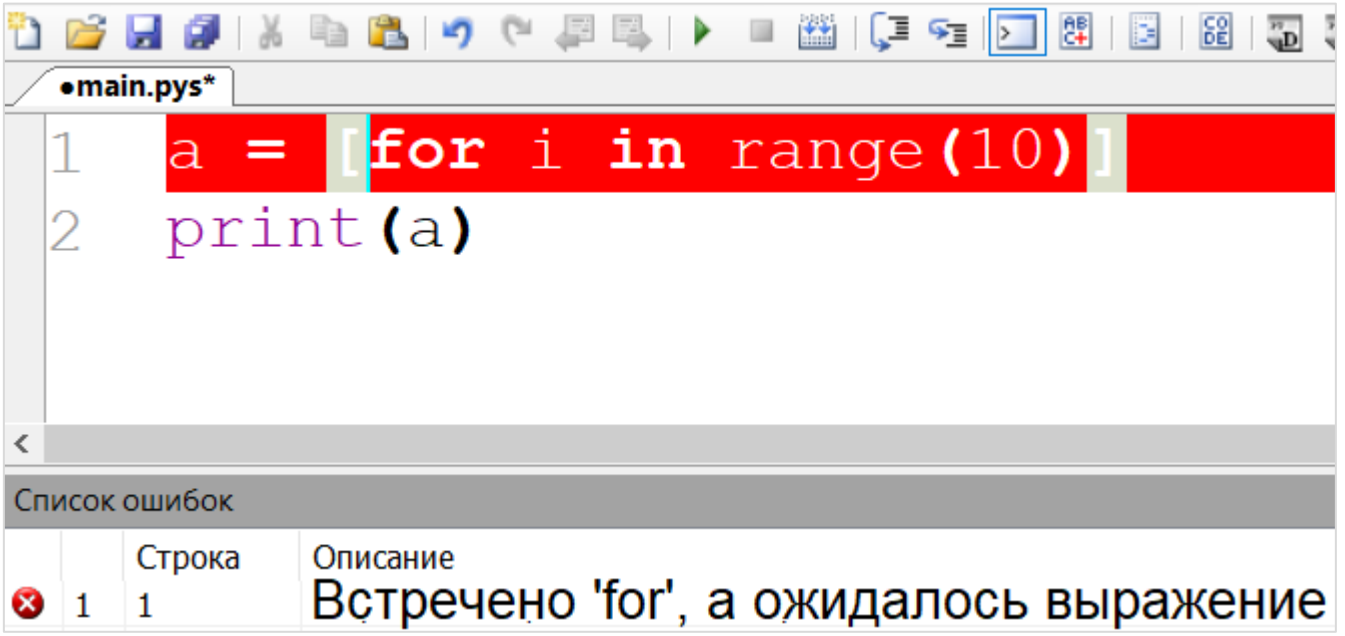
Объявление двумерного списка



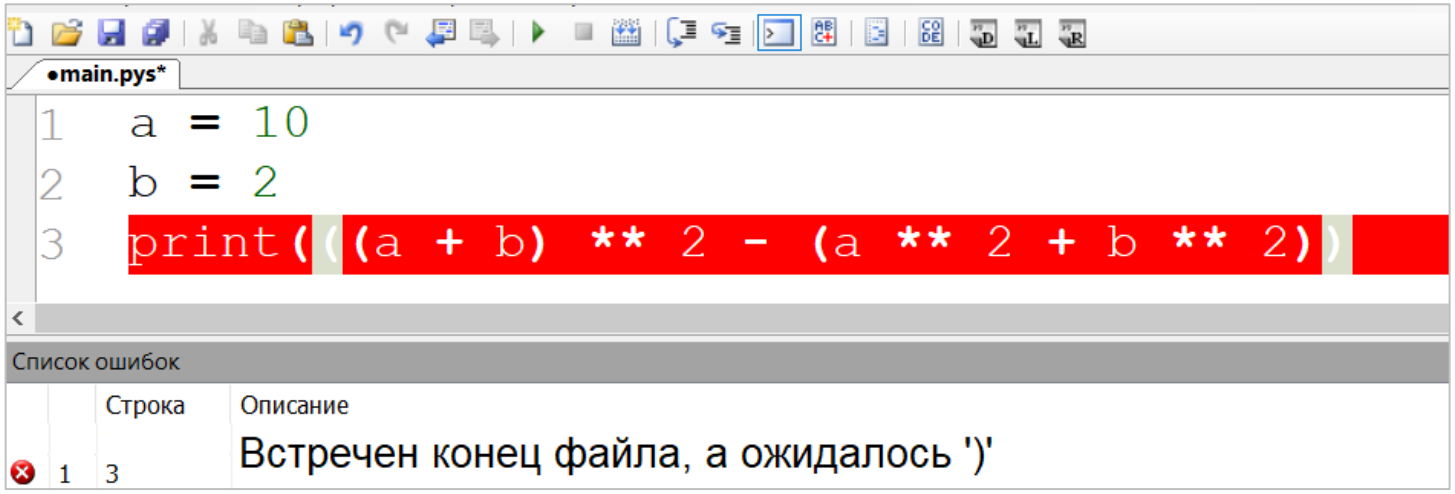
Сообщения об ошибках в SPython

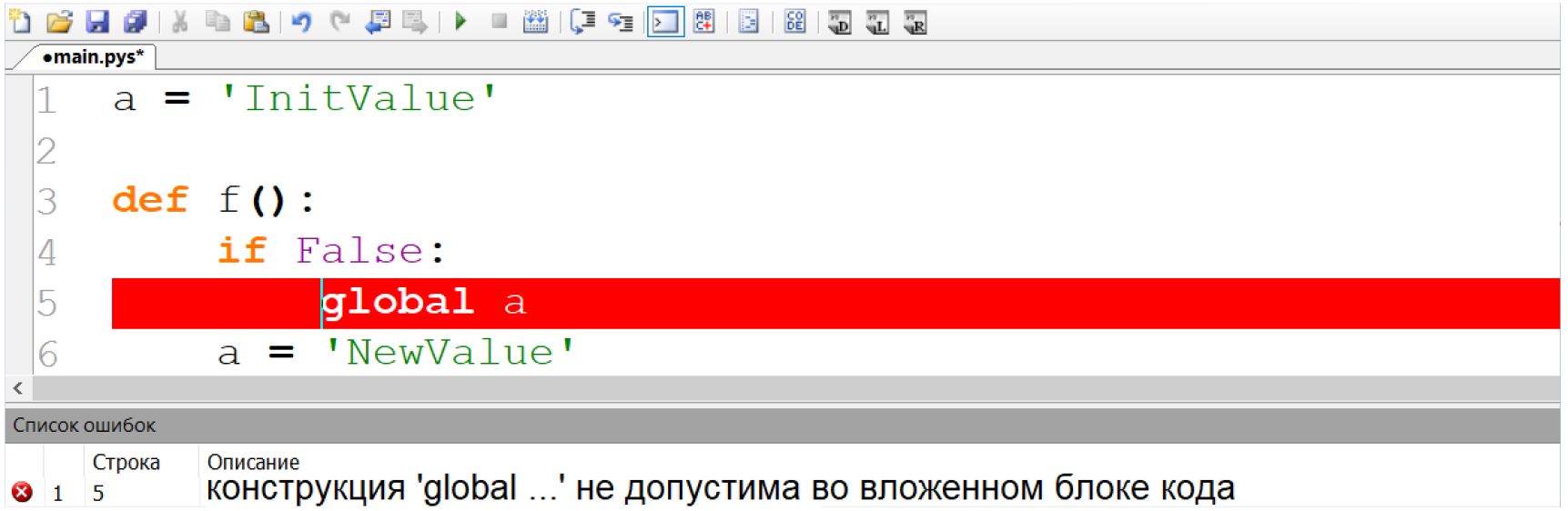
Примеры сообщений об ошибках



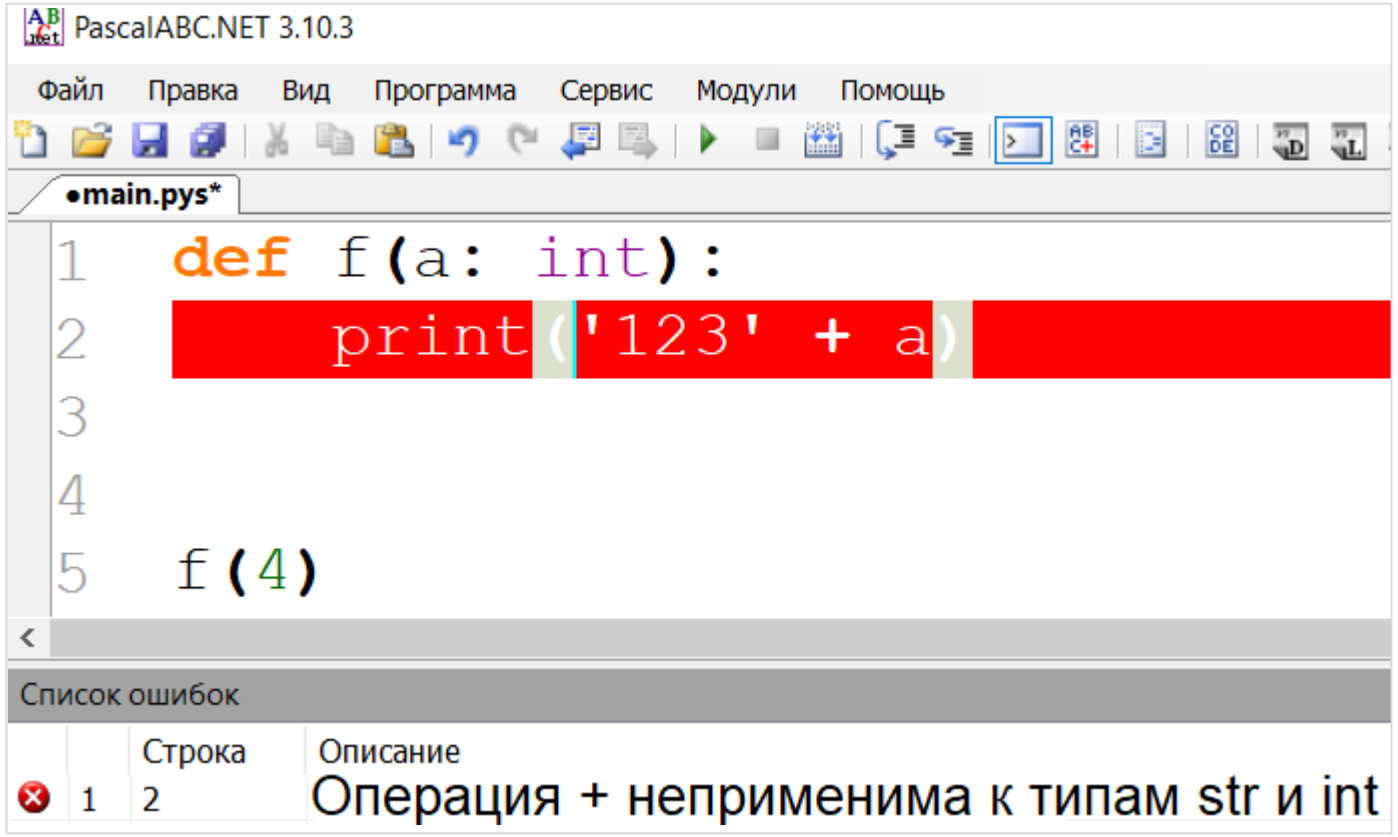


Примеры сообщений об ошибках

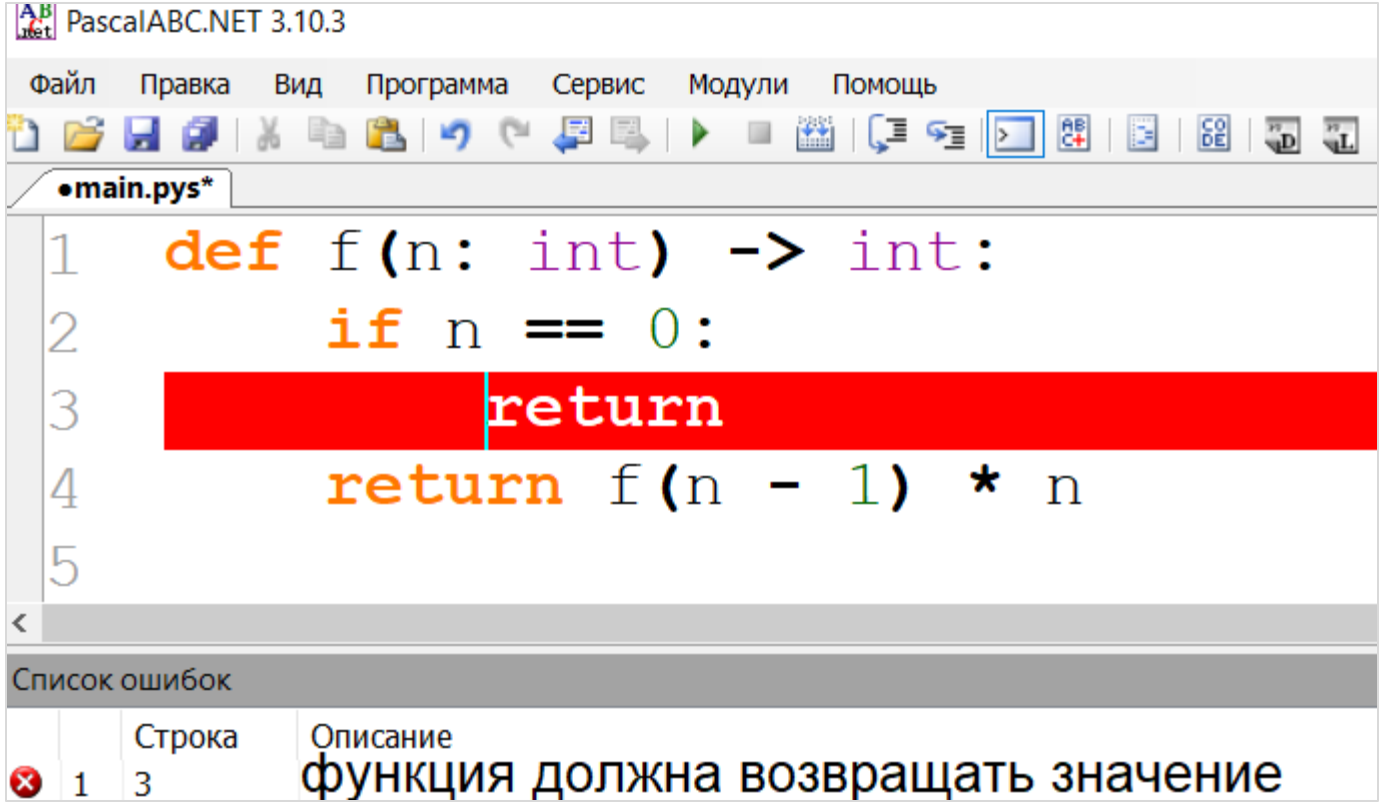




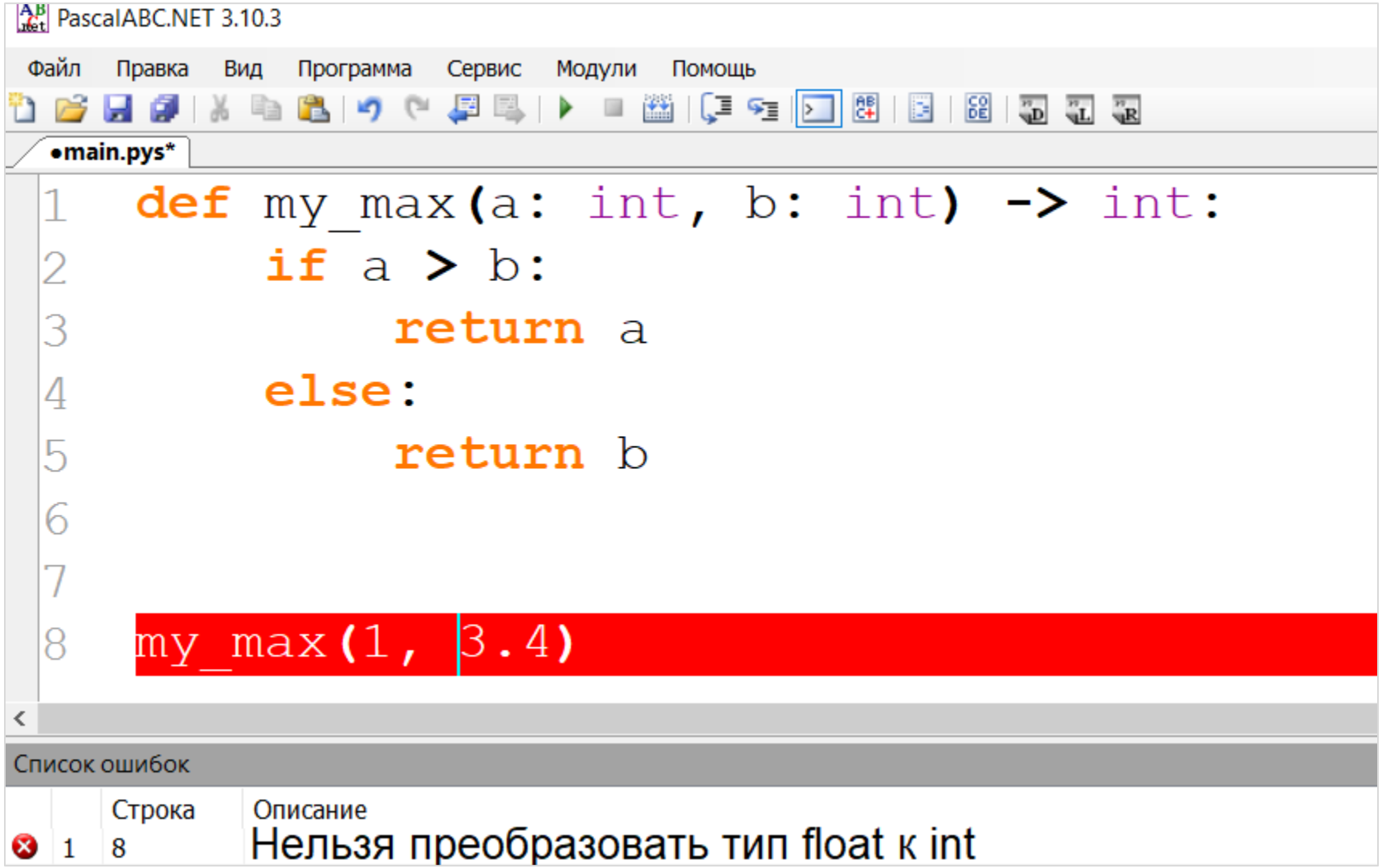
Операция не определена над типами



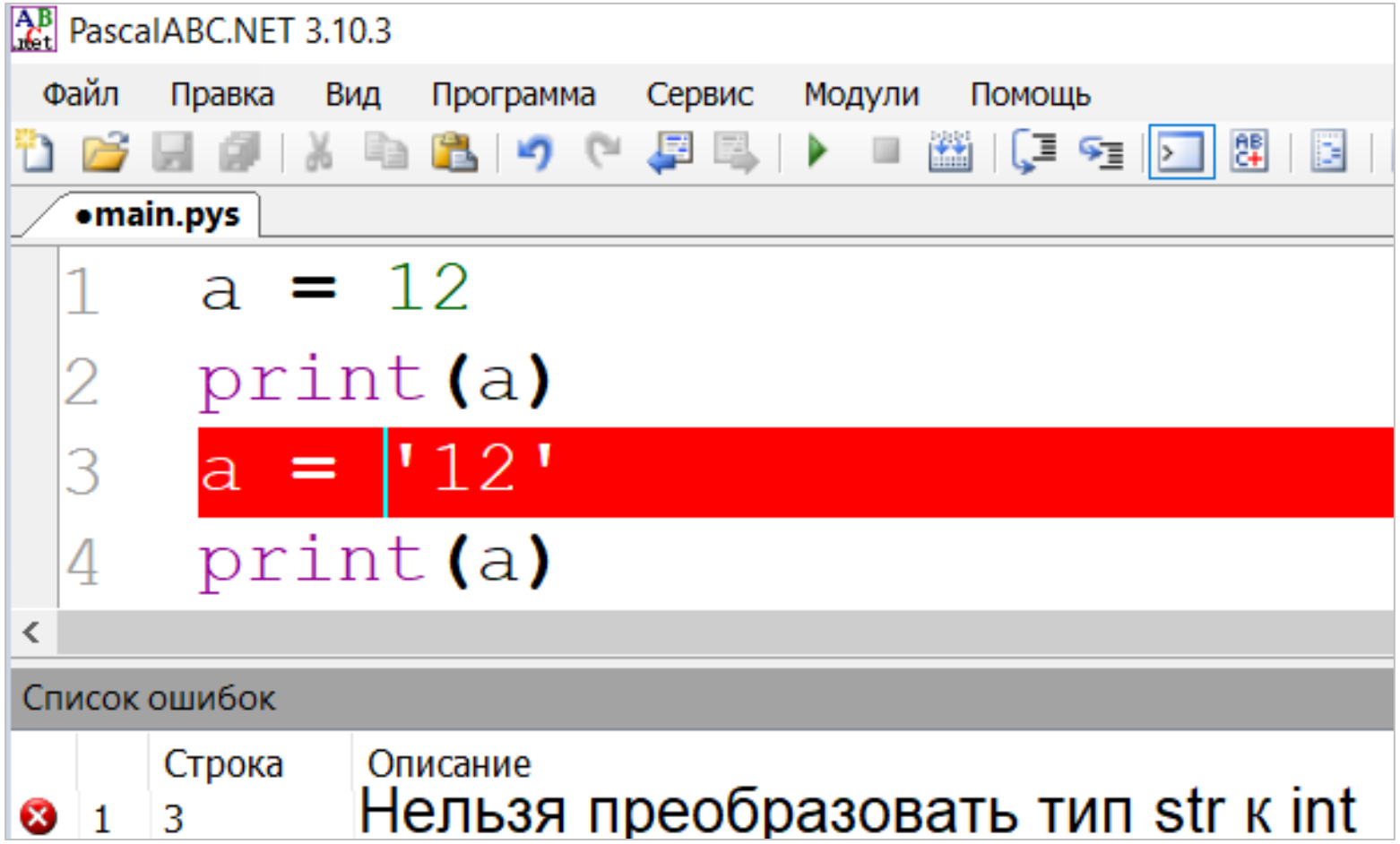
Функция не возвращает значение



Несоответствие типа параметра функции



Несоответствие типа присваиваемого значения



Сравнение инструментов для поиска ошибок

Выявление как можно большего числа ошибок до начала работы программы очень важно для начинающих программистов.

PyCharm выявляет ошибки в основном на этапе выполнения программы, при этом сообщения об ошибках часто бывают слишком общими, например: «invalid syntax».

Компилятор SPython помогает значительно эффективнее: при обнаружении ошибки он укажет не только строку, но и точное место в выражении, где произошла ошибка, а также предоставит максимально подробное сообщение.

Скорость работы программ на SPython

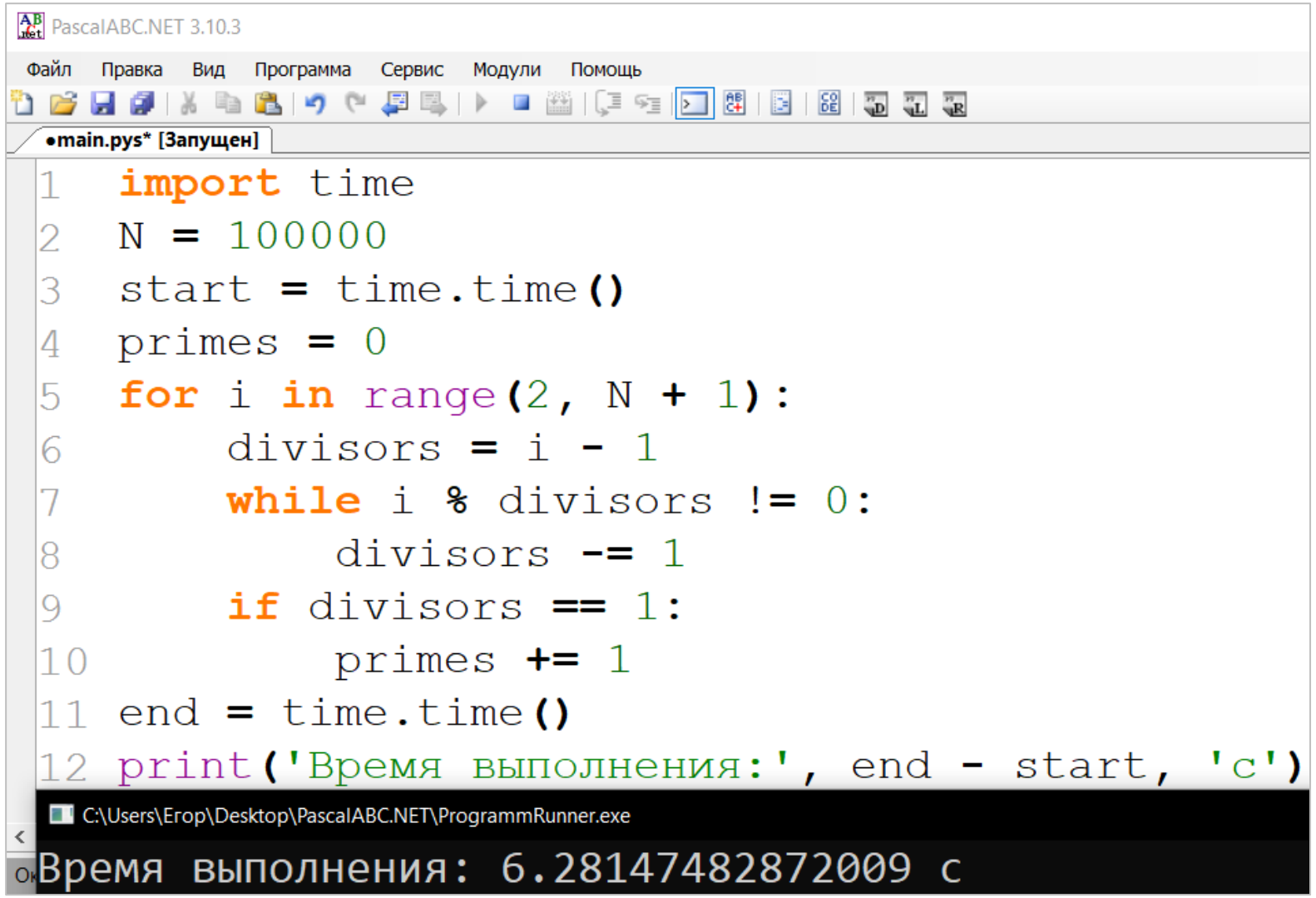
Тестирование времени работы программы

Сравним результаты замеров времени исполнения одинаковых программ на SPython и Python.

В качестве первого примера возьмём программу, наивно считающую количество простых чисел до N.

Для замеров использовался персональный компьютер с процессором
Intel Core i7-8700K CPU 3.70GHz.

Результаты замеров времени: SPython



Результаты замеров времени: Python

main.py

```
1   import time
2   N = 100000
3   start = time.time()
4   primes = 0
5   for i in range(2, N + 1):
6       divisors = i - 1
7       while i % divisors != 0:
8           divisors -= 1
9       if divisors == 1:
10          primes += 1
11  end = time.time()
12  print('Время выполнения:', end - start, 'c')
```

main

```
Время выполнения: 297.94880771636963 c
```

Результаты замеров времени: Python

main.py

```
1   import time
2   N = 100000
3   start = time.time()
4   primes = 0
5   for i in range(2, N + 1):
6       divisors = i - 1
7       while i % divisors != 0:
8           divisors -= 1
9       if divisors == 1:
10          primes += 1
11  end = time.time()
12  print('Время выполнения:', end - start, 'c')
```

main

```
Время выполнения: 297.94880771636963 c
```

В 47 раз медленнее!

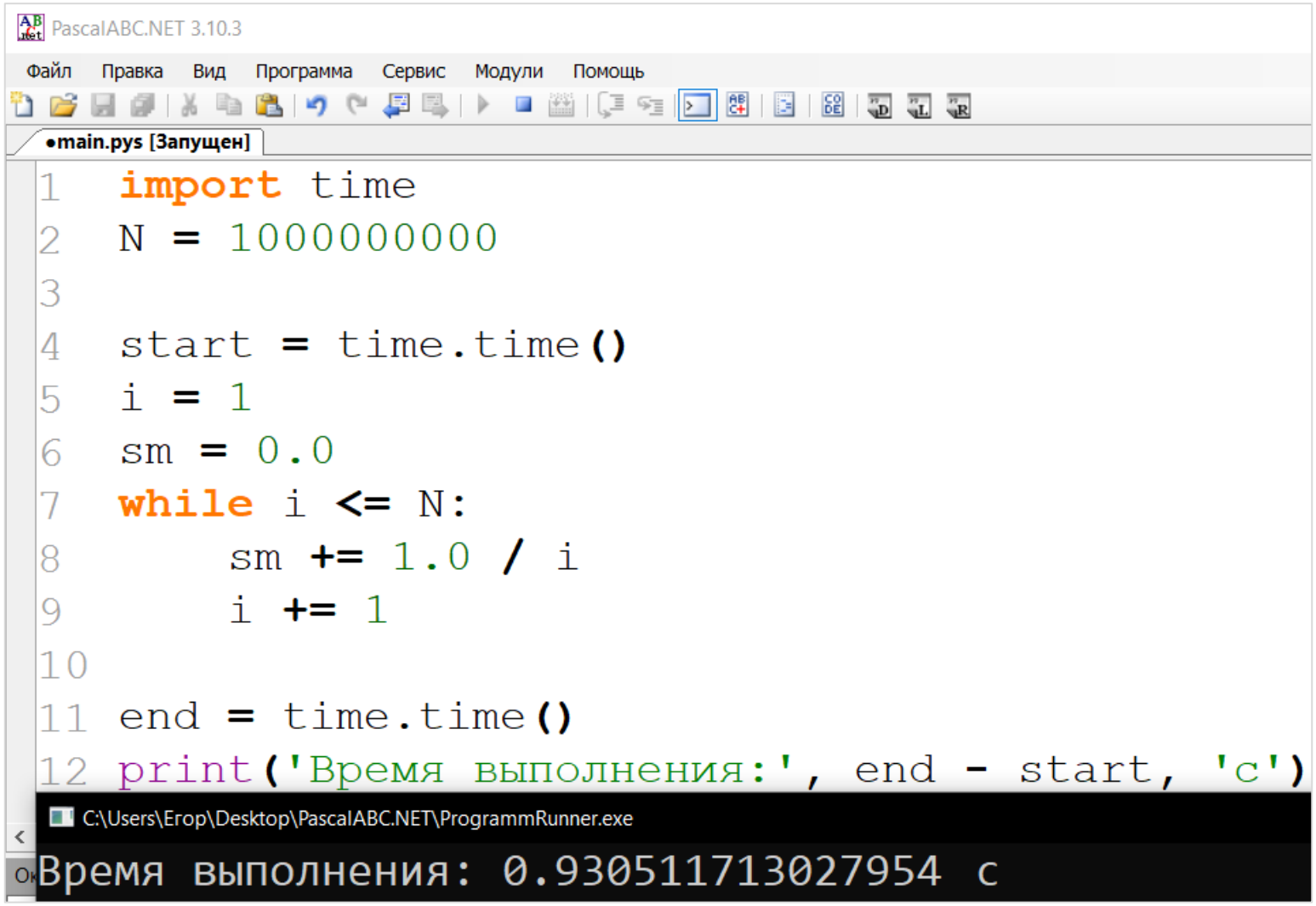
Тестирование времени работы программы

В качестве второго примера возьмём программу, считающую сумму гармонического ряда.

$$H_n = \sum_{k=1}^{n} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}$$

Для замеров использовался персональный компьютер с процессором Intel Core i7-8700K CPU 3.70GHz.

Результаты замеров времени: SPython



Результаты замеров времени: Python

main.py

```
1  import time
2  N = 1000000000
3
4  start = time.time()
5  i = 1
6  sm = 0.0
7  while i <= N:
8      sm += 1.0 / i
9      i += 1
10
11 end = time.time()
12 print('Время выполнения:', end - start, 'c')
```

main

```
Время выполнения: 148.81026244163513 c
```

Результаты замеров времени: Python

main.py

```
1  import time
2  N = 1000000000
3
4  start = time.time()
5  i = 1
6  sm = 0.0
7  while i <= N:
8      sm += 1.0 / i
9      i += 1
10
11 end = time.time()
12 print('Время выполнения:', end - start, 'c')
```

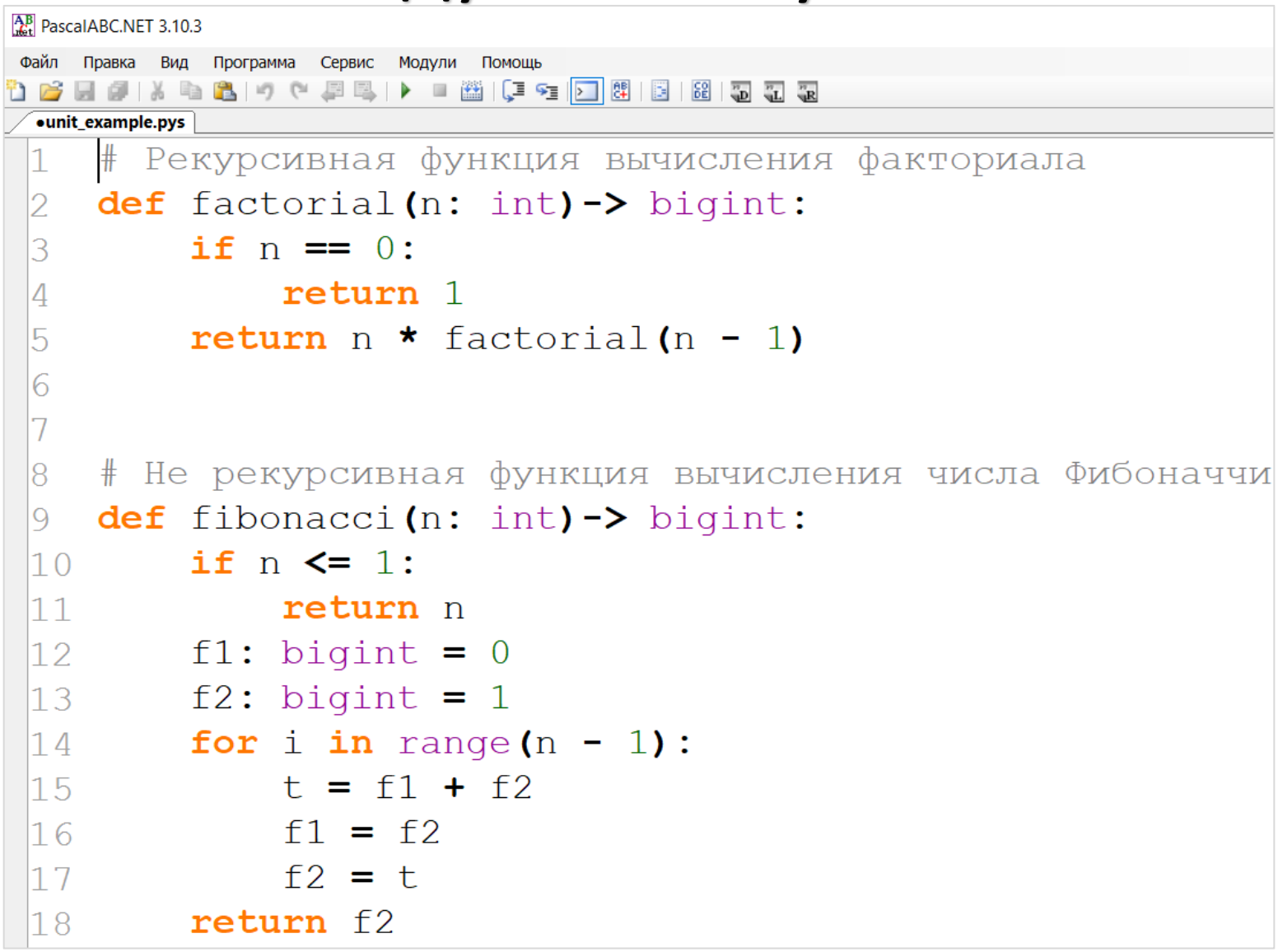
main

```
Время выполнения: 148.81026244163513 c
```

В 160 раз медленнее!

Взаимодействие SPython и PascalABC.NET

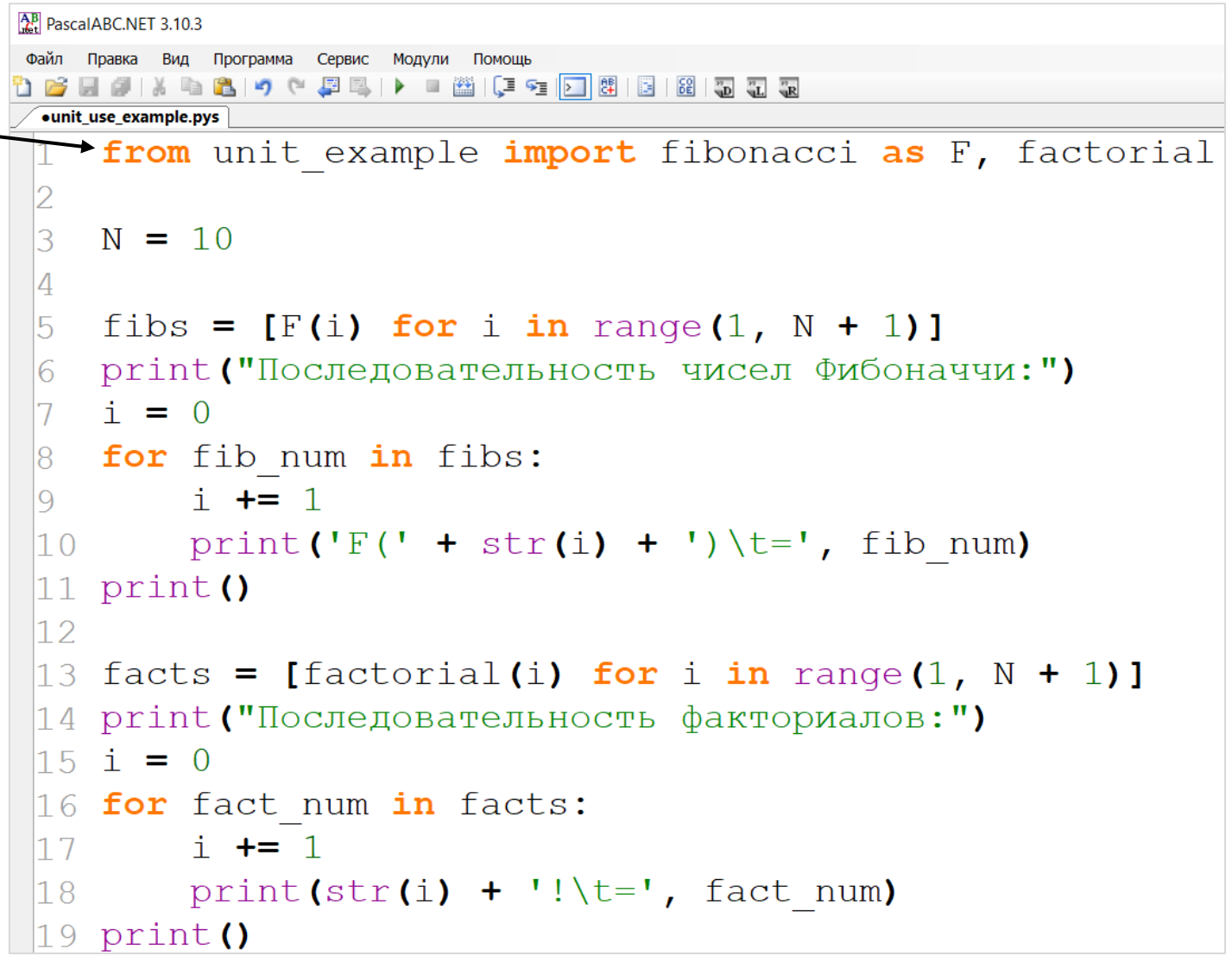
Модуль на SPython




```python
# Рекурсивная функция вычисления факториала
def factorial(n: int)-> bigint:
    if n == 0:
        return 1
    return n * factorial(n - 1)


# Не рекурсивная функция вычисления числа Фибоначчи
def fibonacci(n: int)-> bigint:
    if n <= 1:
        return n
    f1: bigint = 0
    f2: bigint = 1
    for i in range(n - 1):
        t = f1 + f2
        f1 = f2
        f2 = t
    return f2
```

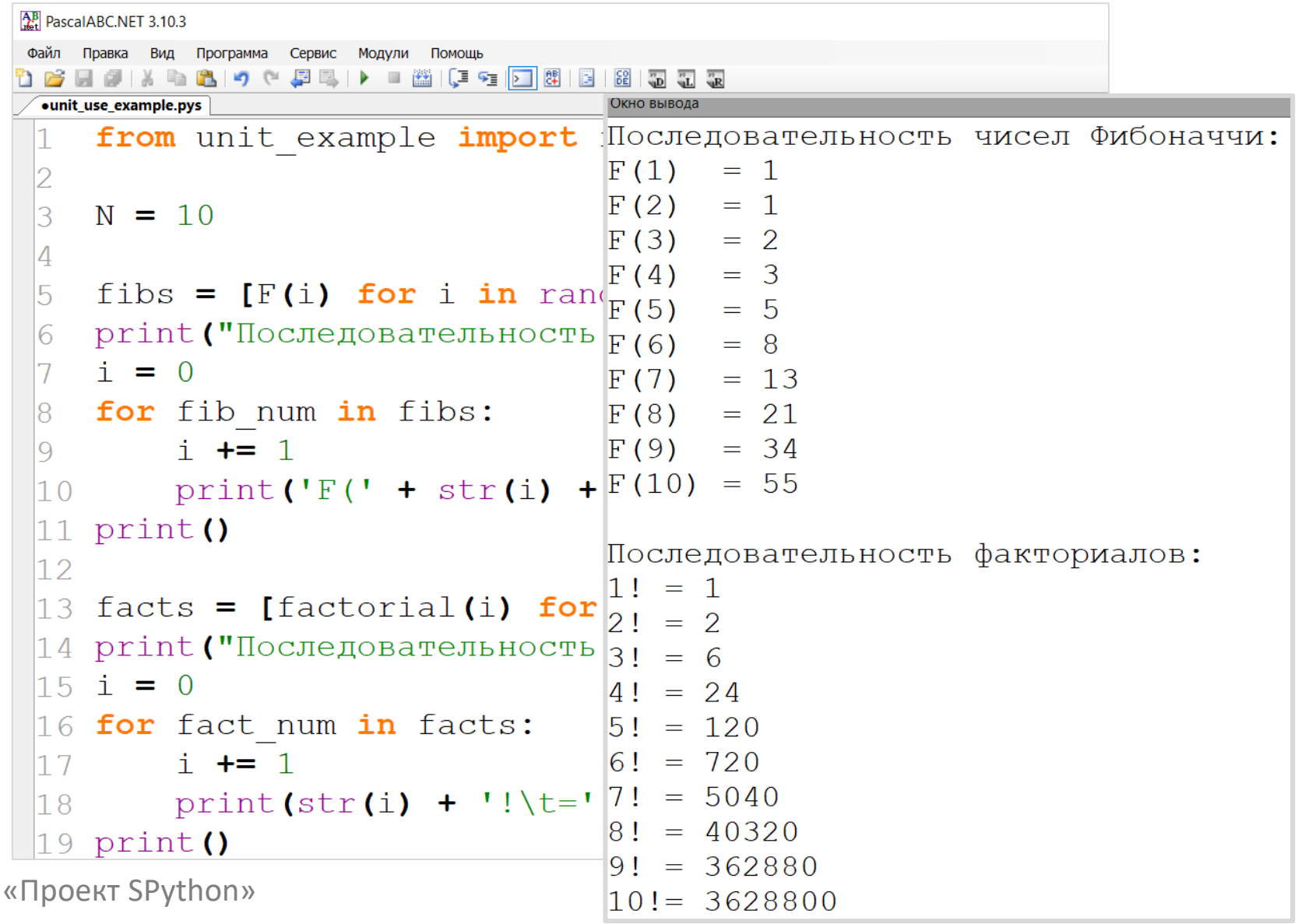
Использование модуля SPython на SPython

Подключение модуля



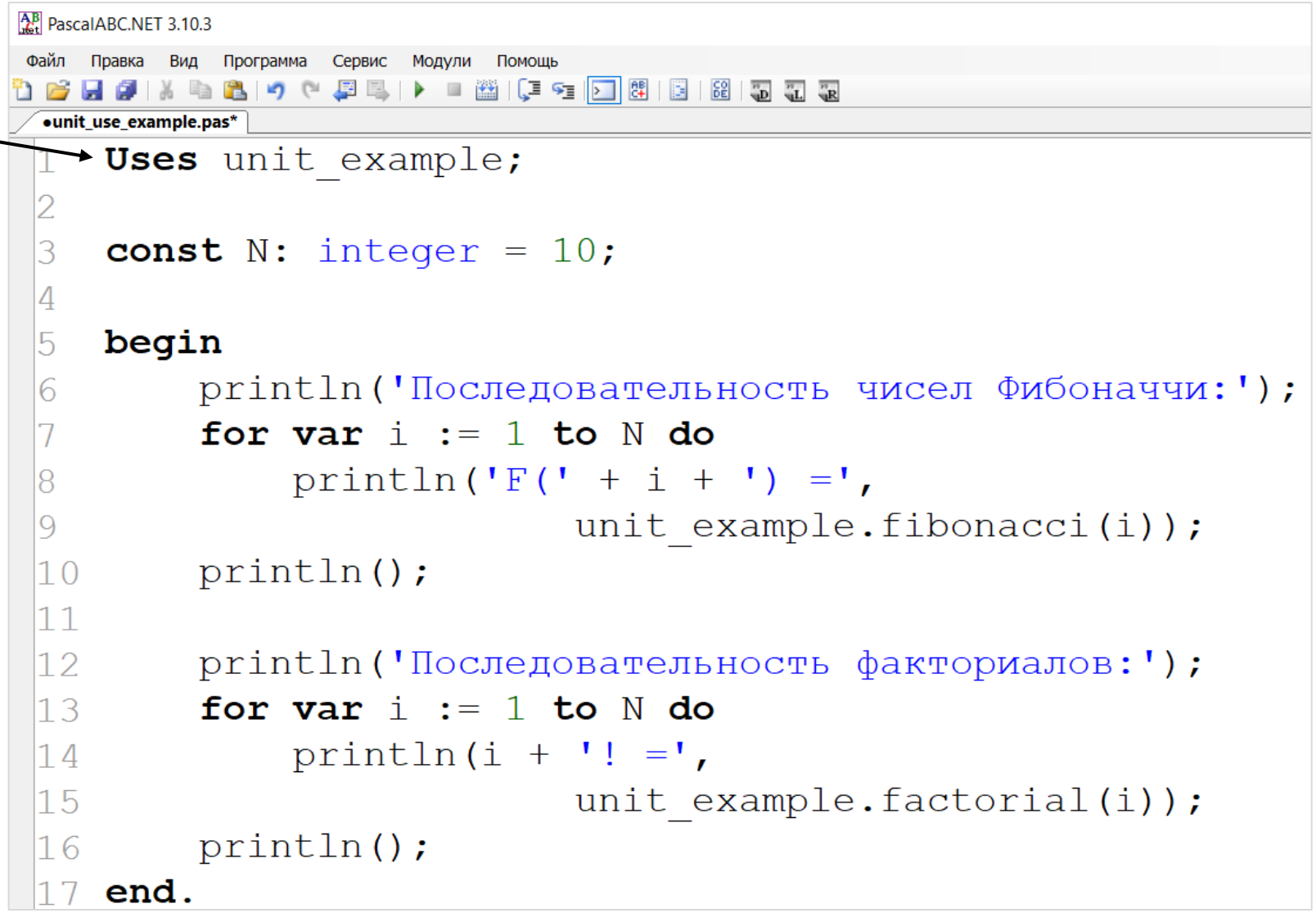

```
from unit_example import fibonacci as F, factorial

N = 10

fibs = [F(i) for i in range(1, N + 1)]
print("Последовательность чисел Фибоначчи:")
i = 0
for fib_num in fibs:
    i += 1
    print('F(' + str(i) + ')\t=', fib_num)
print()

facts = [factorial(i) for i in range(1, N + 1)]
print("Последовательность факториалов:")
i = 0
for fact_num in facts:
    i += 1
    print(str(i) + '!\t=', fact_num)
print()
```

Использование модуля SPython на SPython



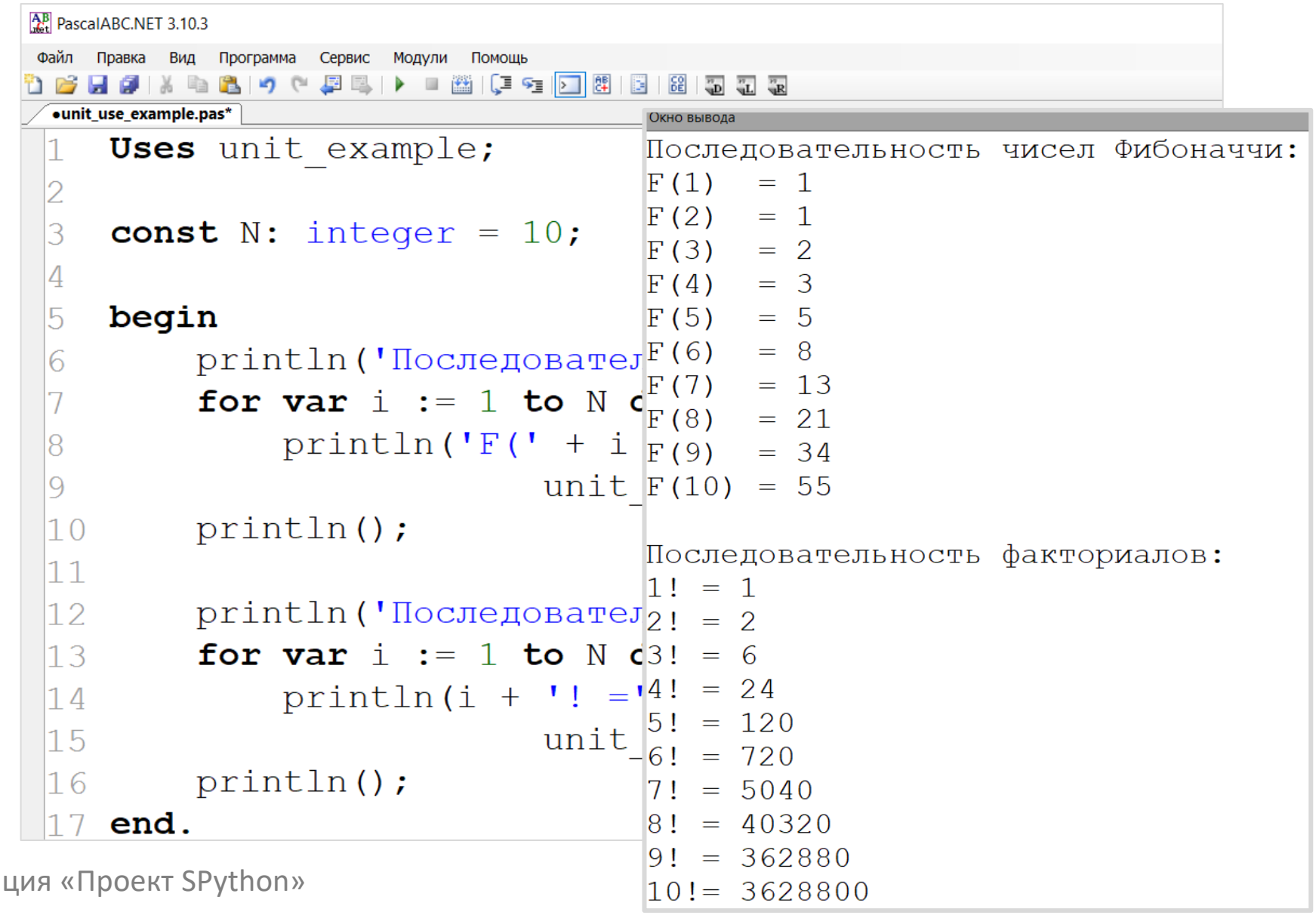
Использование модуля SPython на PascalABC.NET

Подключение модуля

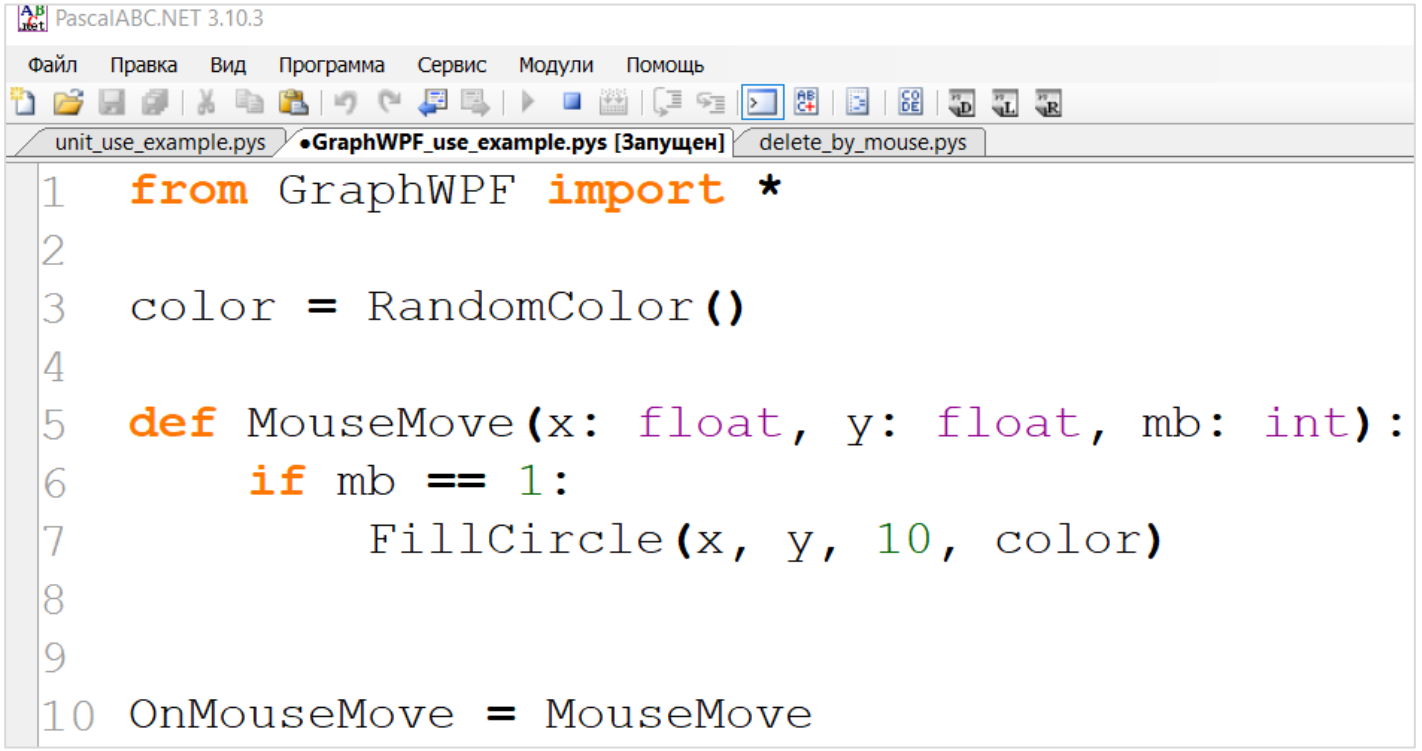



```
Uses unit_example;

const N: integer = 10;

begin
    println('Последовательность чисел Фибоначчи:');
    for var i := 1 to N do
        println('F(' + i + ') =',
                unit_example.fibonacci(i));
    println();

    println('Последовательность факториалов:');
    for var i := 1 to N do
        println(i + '! =',
                unit_example.factorial(i));
    println();
end.
```

Использование модуля SPython на PascalABC.NET



Использование модуля GraphWPF на SPython



Использование модуля GraphWPF на SPython

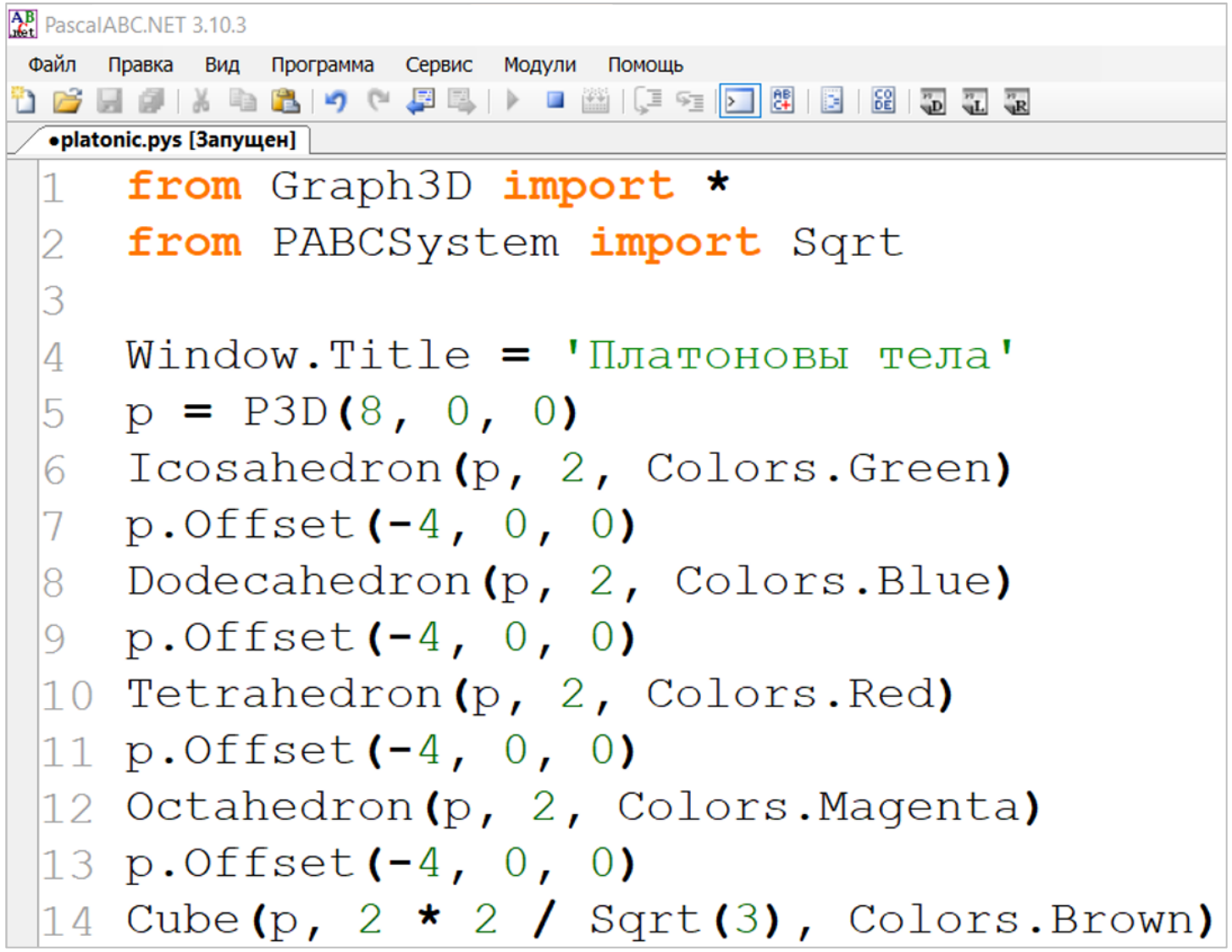
PascalABC.NET 3.10.3

Файл Правка Вид Программа Сервис Модули Помощь

unit_use_example.pys | •GraphWPF_use_example.pys [Запущен] | delete_by_mouse.pys

```
1  from GraphWPF import *
2
3  color = RandomColor()
4
5  def MouseMove(x: float, y: float, mb: int):
6      if mb == 1:
7          FillCircle(x
8
9
10 OnMouseMove = MouseM
```

Использование модуля Graph3D на SPython




```
1  from Graph3D import *
2  from PABCSystem import Sqrt
3
4  Window.Title = 'Платоновы тела'
5  p = P3D(8, 0, 0)
6  Icosahedron(p, 2, Colors.Green)
7  p.Offset(-4, 0, 0)
8  Dodecahedron(p, 2, Colors.Blue)
9  p.Offset(-4, 0, 0)
10 Tetrahedron(p, 2, Colors.Red)
11 p.Offset(-4, 0, 0)
12 Octahedron(p, 2, Colors.Magenta)
13 p.Offset(-4, 0, 0)
14 Cube(p, 2 * 2 / Sqrt(3), Colors.Brown)
```

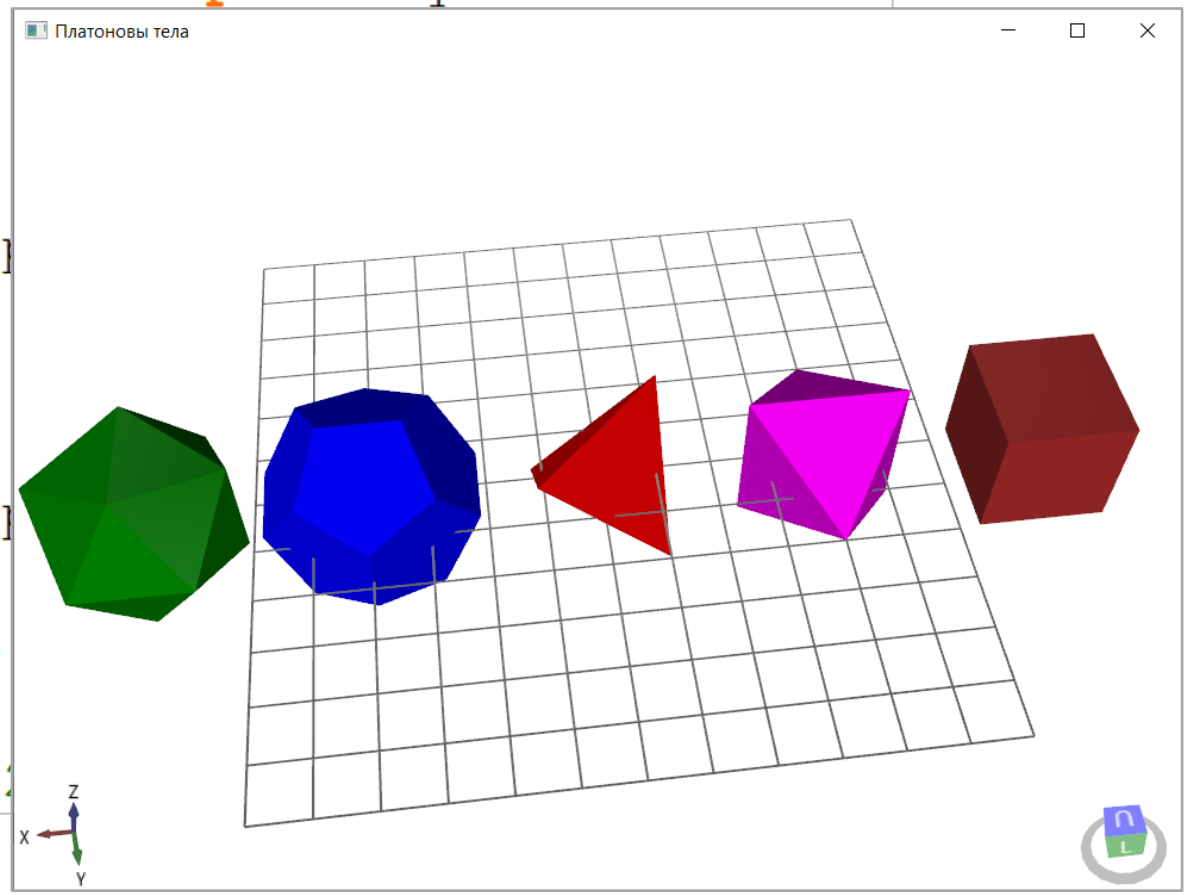
Использование модуля Graph3D на SPython

PascalABC.NET 3.10.3

Файл Правка Вид Программа Сервис Модули Помощь

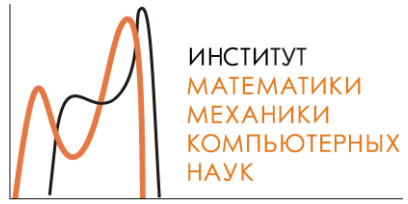
•platonic.pys [Запущен]

```
1  from Graph3D import *
2  from PABCSystem import Sqrt
3
4  Window.Title
5  p = P3D(8, 0
6  Icosahedron(
7  p.Offset(-4,
8  Dodecahedron
9  p.Offset(-4,
10 Tetrahedron(
11 p.Offset(-4,
12 Octahedron(p
13 p.Offset(-4,
14 Cube(p, 2 *
```



Установка SPython

1. Переходим на github.com/AlexanderZemlyak/pascalabcnet/releases/tag/SPython0.0.1
2. Скачиваем SPython.zip
3. В свойствах файла устанавливаем флаг «Разблокировать»
4. Разархивируем SPython.zip
5. Запускаем PascalABCNET.exe
6. Пользуемся примерами из папки SPythonExamples



Мовчан Е.В.

Южный федеральный университет,
Институт математики, механики и компьютерных наук

emovchan@sfedu.ru