

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?



+



=

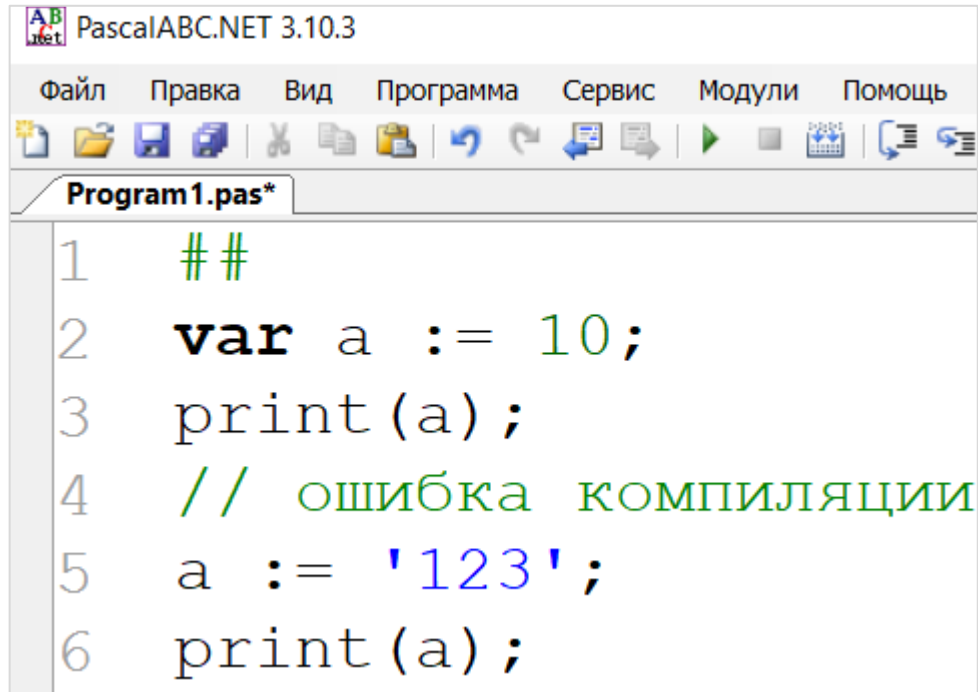
SPython

Основные понятия

Виды типизации

Статическая типизация:

у переменной неизменяемый тип,
известный на этапе компиляции

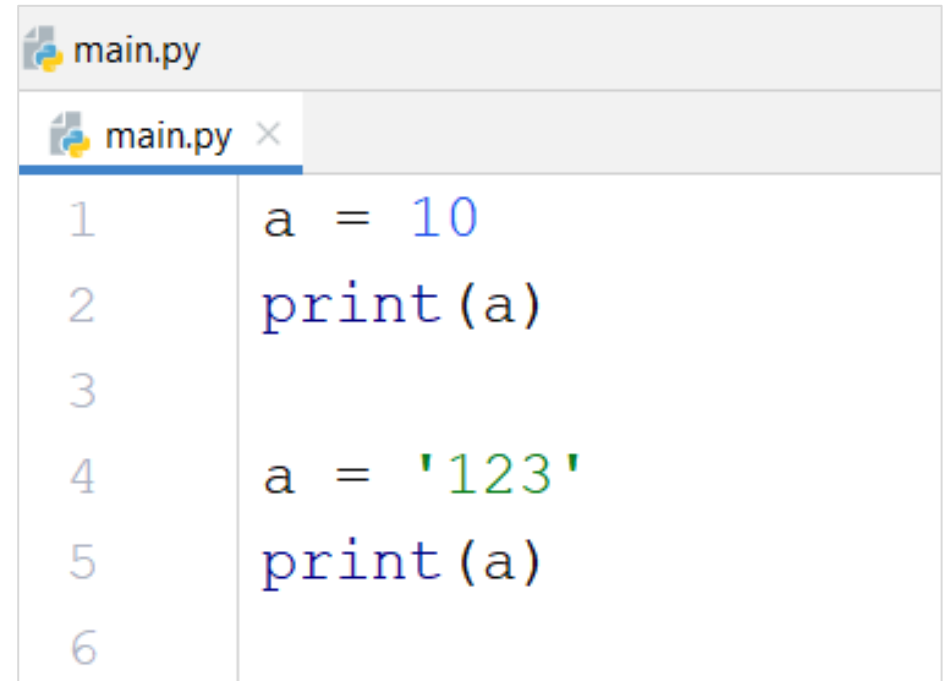


The screenshot shows the PascalABC.NET 3.10.3 IDE. The menu bar includes 'Файл', 'Правка', 'Вид', 'Программа', 'Сервис', 'Модули', and 'Помощь'. The toolbar contains various icons for file operations and execution. The active window is 'Program1.pas*'. The code is as follows:

```
1  ##
2  var a := 10;
3  print(a);
4  // ошибка компиляции
5  a := '123';
6  print(a);
```

Динамическая типизация:

тип переменной может меняться в
процессе выполнения программы



The screenshot shows a Python IDE with a window titled 'main.py'. The code is as follows:

```
1  a = 10
2  print(a)
3
4  a = '123'
5  print(a)
6
```

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

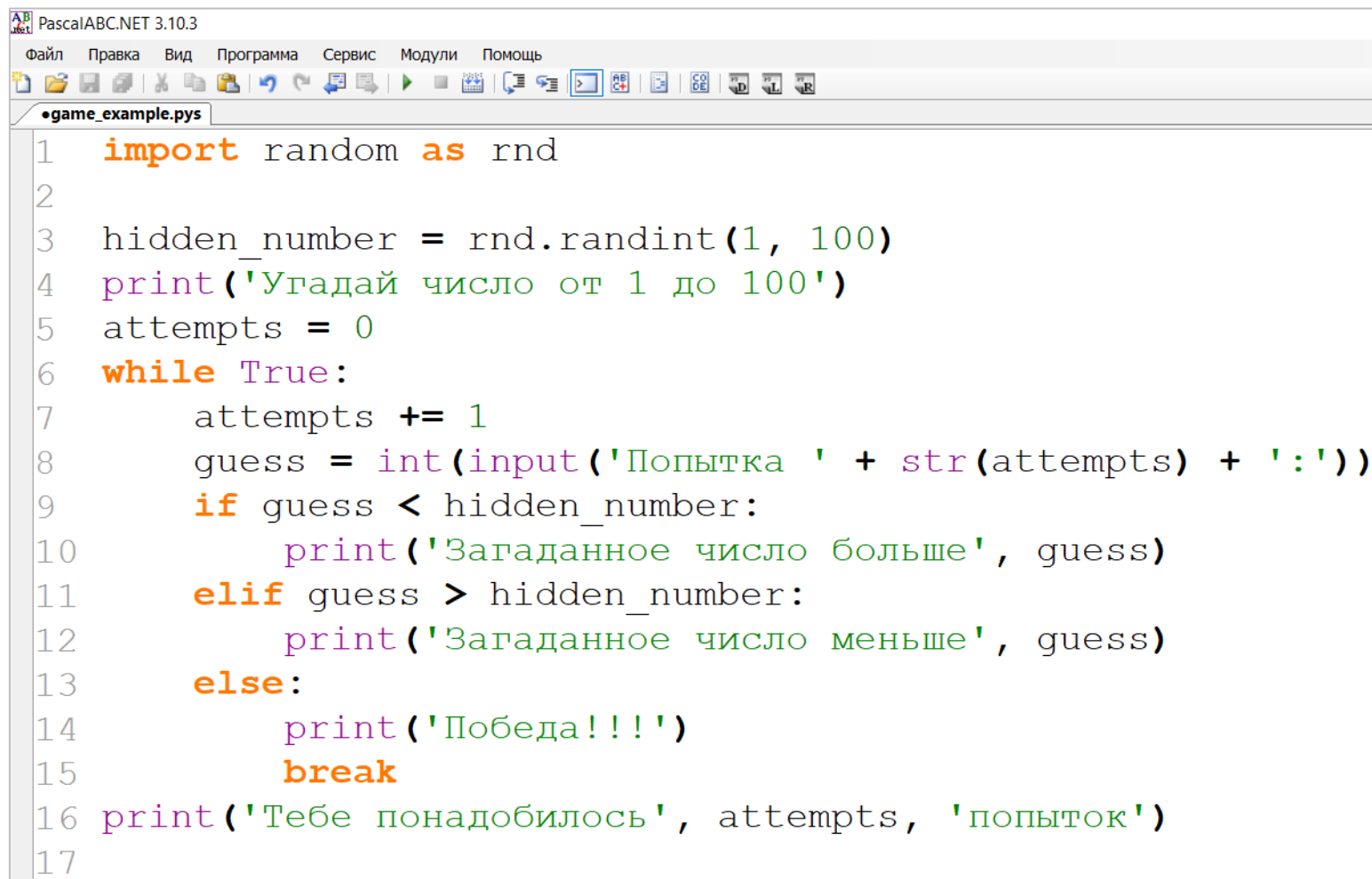
- Высокая производительность кода
- Раннее обнаружение ошибок
- Реализация огромного спектра оптимизаций
- Более строгие инструменты разработки
- Поддержка статического анализа кода
- Гарантии корректности скомпилированного кода

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



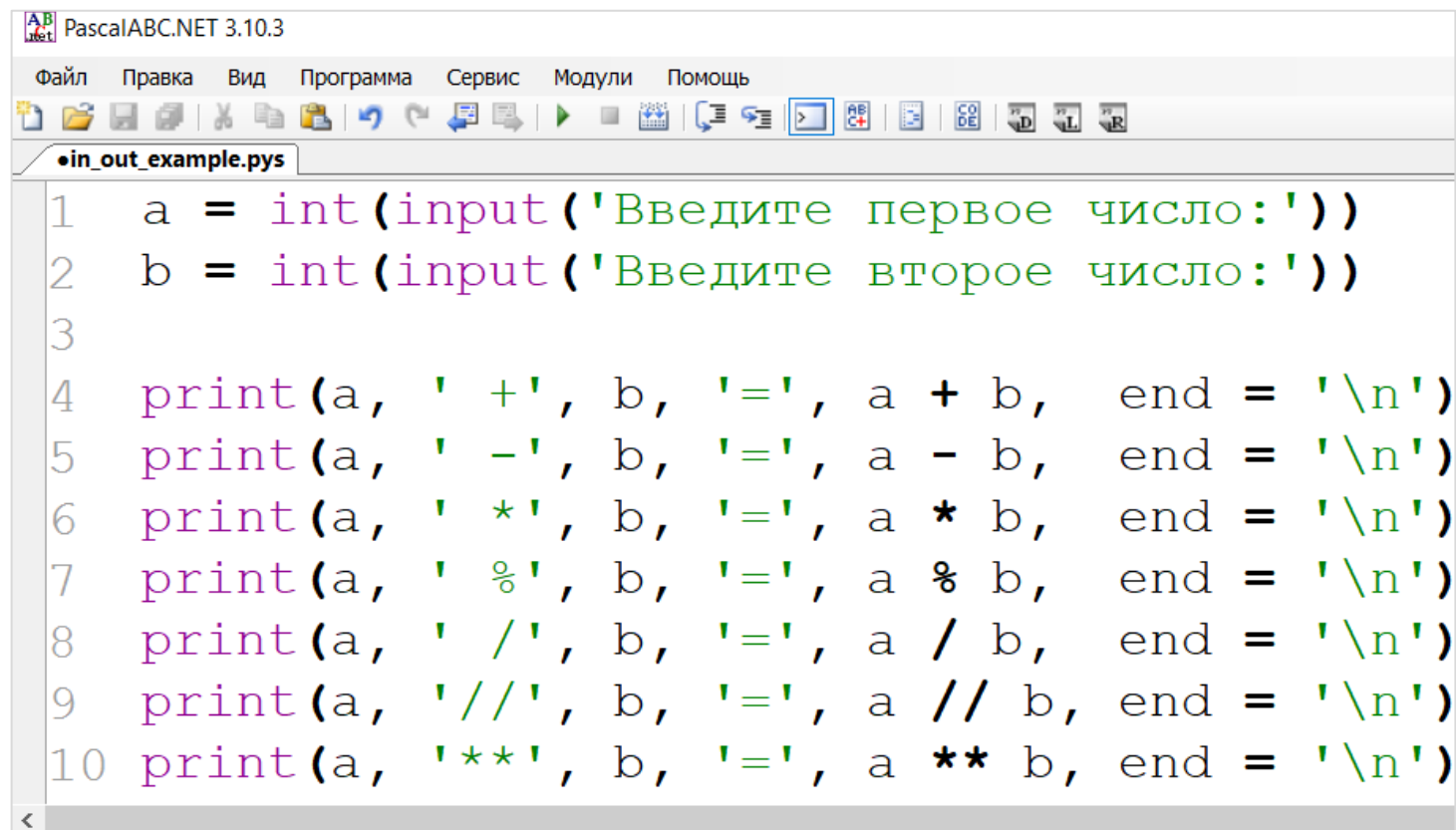
```
PascalABC.NET 3.10.3
Файл  Правка  Вид  Программа  Сервис  Модули  Помощь
•game_example.py
1  import random as rnd
2
3  hidden_number = rnd.randint(1, 100)
4  print('Угадай число от 1 до 100')
5  attempts = 0
6  while True:
7      attempts += 1
8      guess = int(input('Попытка ' + str(attempts) + ':'))
9      if guess < hidden_number:
10         print('Загаданное число больше', guess)
11     elif guess > hidden_number:
12         print('Загаданное число меньше', guess)
13     else:
14         print('Победа!!!')
15         break
16 print('Тебе понадобилось', attempts, 'попыток')
17
```

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



The screenshot shows the PascalABC.NET 3.10.3 IDE. The menu bar includes 'Файл', 'Правка', 'Вид', 'Программа', 'Сервис', 'Модули', and 'Помощь'. The toolbar contains various icons for file operations, editing, and execution. The active window is 'in_out_example.pys'. The code is as follows:

```
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

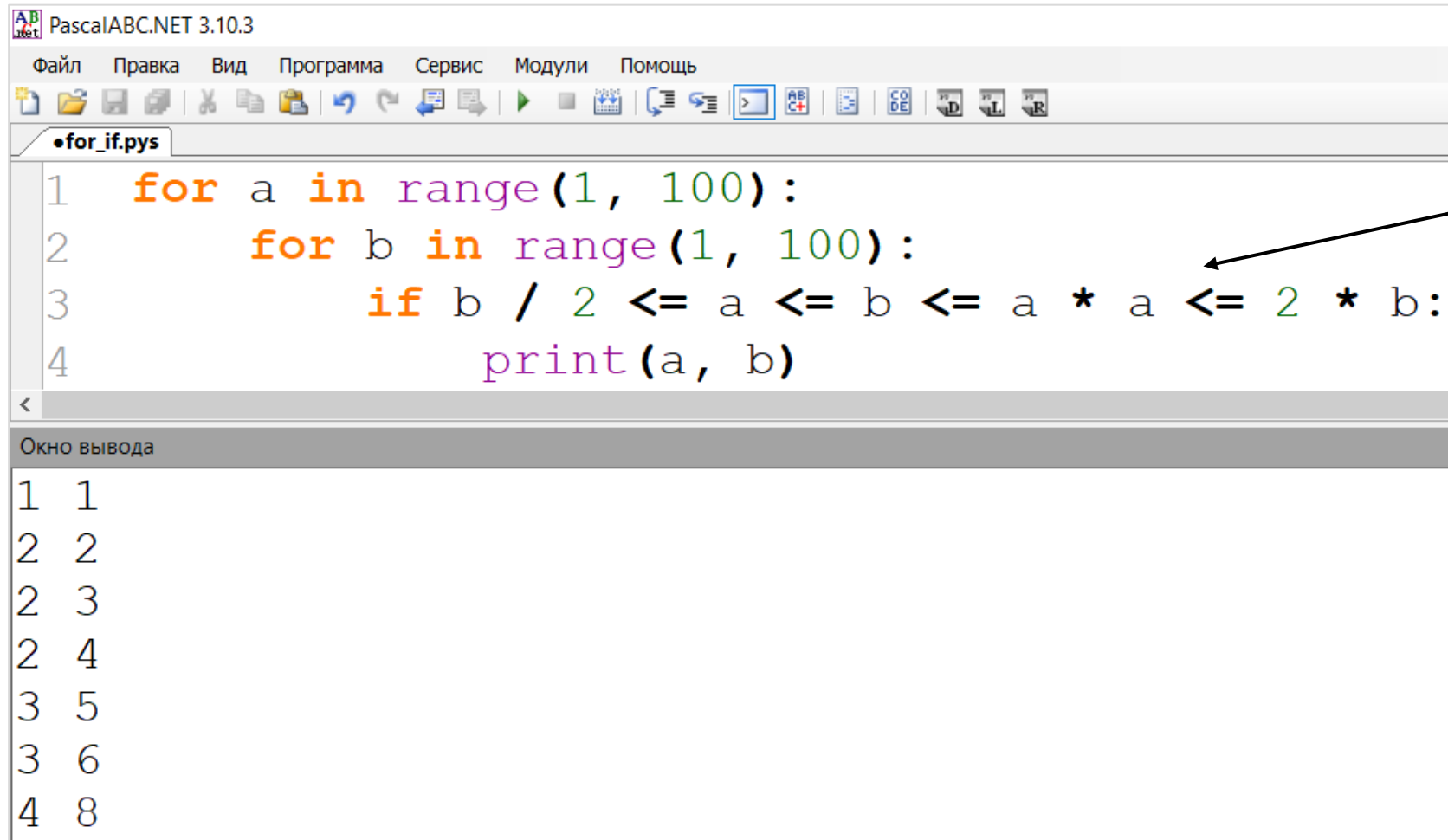
```
PascalABC.NET 3.10.3
Файл  Правка  Вид  Программа  Сервис  Модули  Помощь
in_out_example.pys
1  a = int(input('Введите первое число:'))
2  b = int(input('Введите второе число:'))
3
4  print(a, ' + ', b, '=', a + b)
5  print(a, ' - ', b, '=', a - b)
6  print(a, ' * ', b, '=', a * b)
7  print(a, '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

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



The screenshot shows the PascalABC.NET 3.10.3 IDE. The main window displays a Python script named `for_if.pys` with the following code:

```
1  for a in range(1, 100):  
2      for b in range(1, 100):  
3          if b / 2 <= a <= b <= a * a <= 2 * b:  
4              print(a, b)
```

Below the code editor is a window titled "Окно вывода" (Output Window) showing the results of the program execution:

```
1 1  
2 2  
2 3  
2 4  
3 5  
3 6  
4 8
```

Цепочечные
условия

Поиск индекса минимального элемента в списке

Объявление
функции

Тип списка

Генератор списка

```
PascalABC.NET 3.10.3
Файл  Правка  Вид  Программа  Сервис  Модули  Помощь
•argmin.pys
1  # индекс минимального элемента
2  def argmin(arr: list[int]) -> int:
3      if len(arr) == 0:
4          return -1
5      res = 0
6      for i in range(1, len(arr)):
7          if arr[i] < arr[res]:
8              res = i
9      return res
10
11 # a = [4, 9, 16, 8, 2, 15, 13, 13, 15, 2, 8, 16, 9, 4]
12 a = [i ** 2 % 17 for i in range(2, 16)]
13 print(argmin(a))
<
Окно вывода
4
```

Работа с двумерными списками

Объявление
двумерного
списка

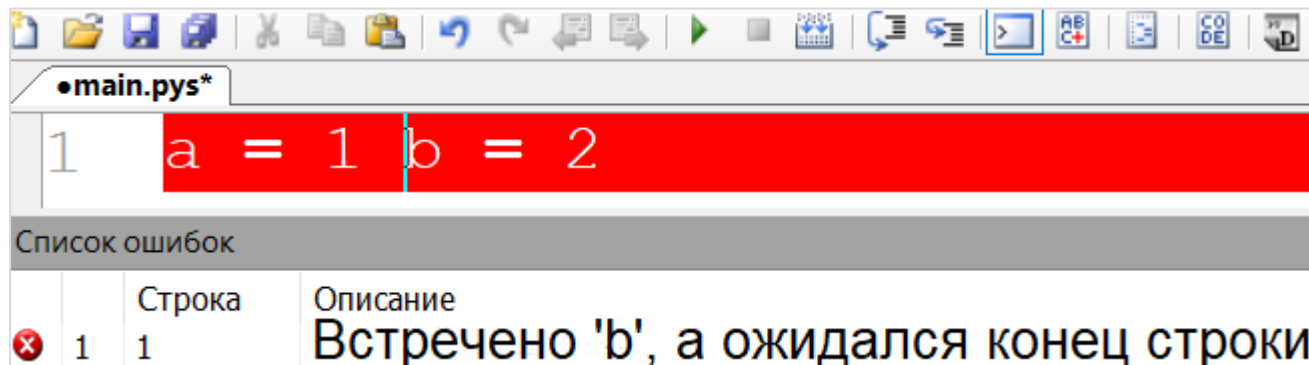
```
PascalABC.NET 3.10.3
Файл  Правка  Вид  Программа  Сервис  Модули  Помощь
list_use_example.pys
1  def print_matr(matr: list[list[int]]):
2      for row in matr:
3          for elem in row:
4              print(elem, '\t', end=' ')
5              print()
6
7
8  matrix_of_integers: list[list[int]]
9  matrix_of_integers = [
10      [j * i for j in range(1, 10)]
11      for i in range(1, 10) if i % 2 == 0
12  ]
13 print_matr(matrix_of_integers)
```

Окно вывода

2	4	6	8	10	12	14	16	18
4	8	12	16	20	24	28	32	36
6	12	18	24	30	36	42	48	54
8	16	24	32	40	48	56	64	72

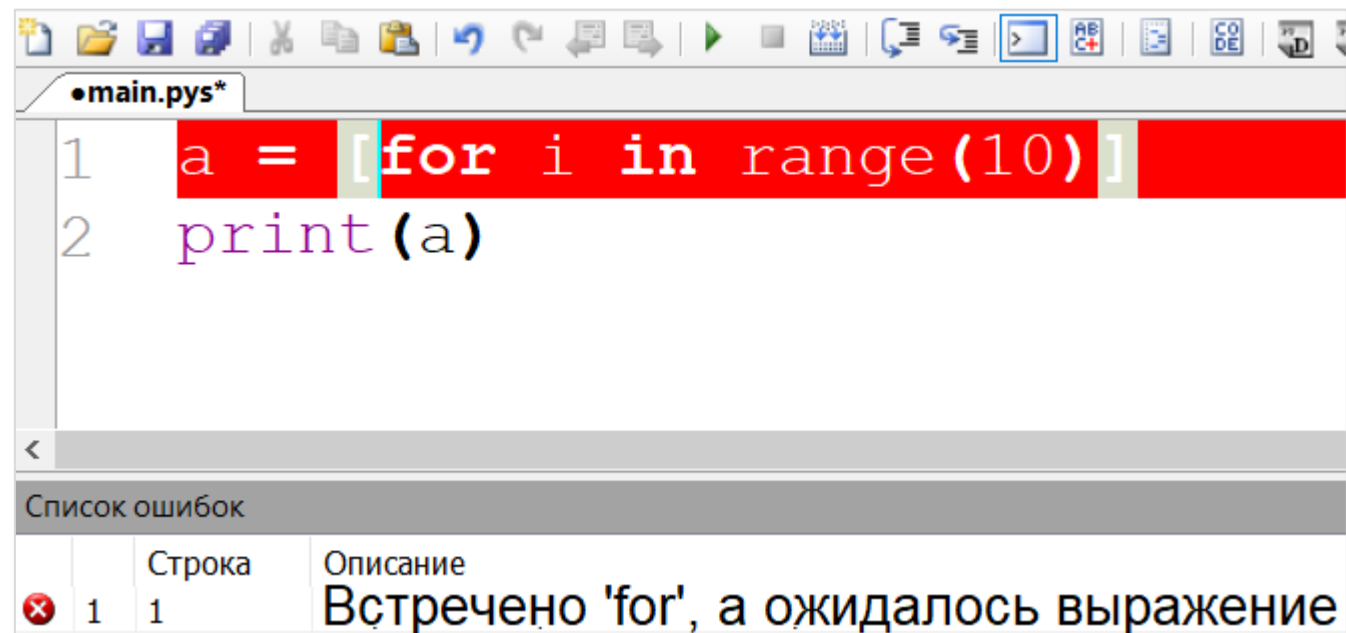
Сообщения об ошибках в SPython

Примеры сообщений об ошибках



```
1 a = 1 b = 2
```

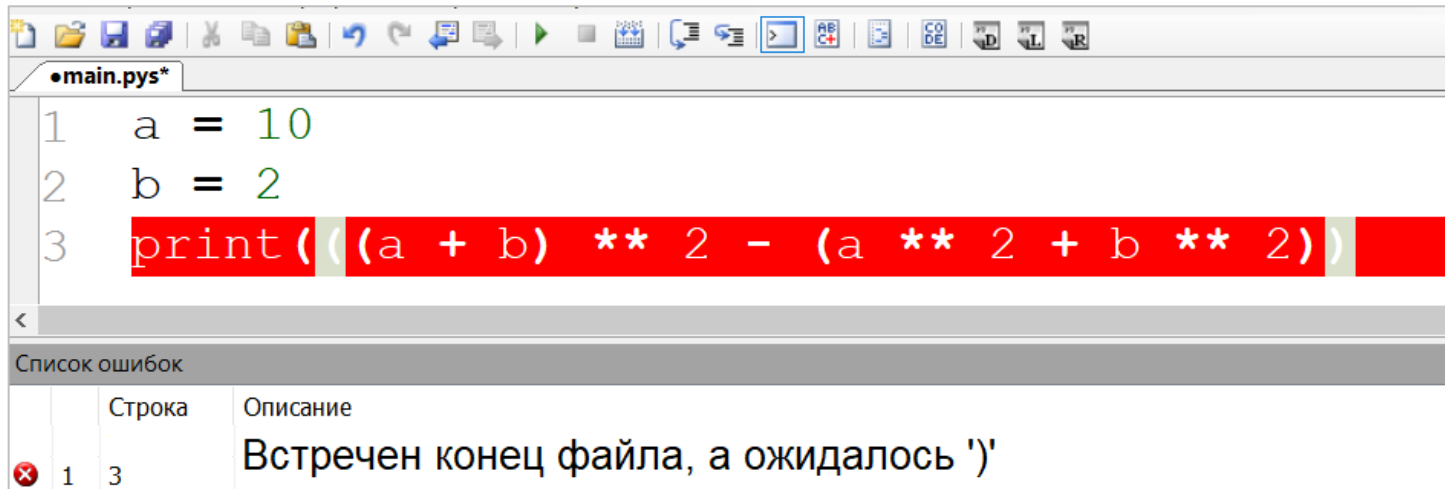
Строка	Описание
1	Встречено 'b', а ожидался конец строки



```
1 a = [for i in range(10)]  
2 print(a)
```

Строка	Описание
1	Встречено 'for', а ожидалось выражение

Примеры сообщений об ошибках

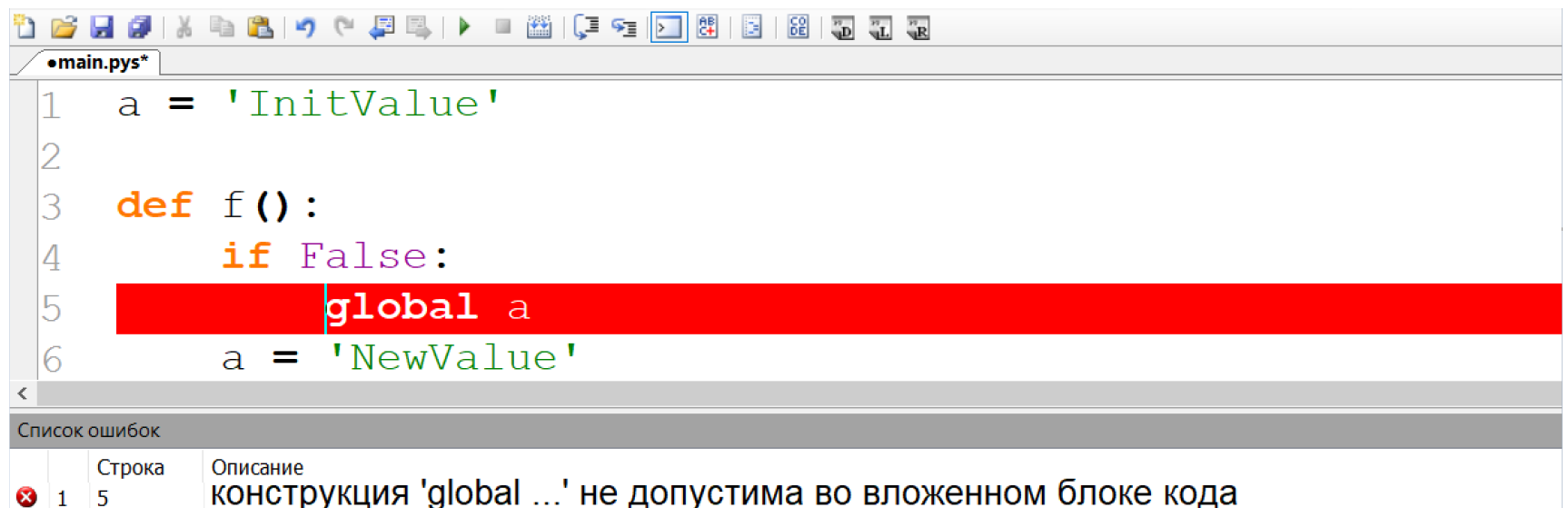


The screenshot shows a Python IDE window titled "main.pys*". The code editor contains three lines of Python code:

```
1 a = 10
2 b = 2
3 print((a + b) ** 2 - (a ** 2 + b ** 2))
```

The third line is highlighted in red, indicating a syntax error. Below the code editor is a table titled "Список ошибок" (Error List).

	Строка	Описание
✖ 1	3	Встречен конец файла, а ожидалось ')'



The screenshot shows a Python IDE window titled "main.pys*". The code editor contains six lines of Python code:

```
1 a = 'InitValue'
2
3 def f():
4     if False:
5         global a
6         a = 'NewValue'
```

The fifth line is highlighted in red, indicating a syntax error. Below the code editor is a table titled "Список ошибок" (Error List).

	Строка	Описание
✖ 1	5	конструкция 'global ...' не допустима во вложенном блоке кода

Операция не определена над типами

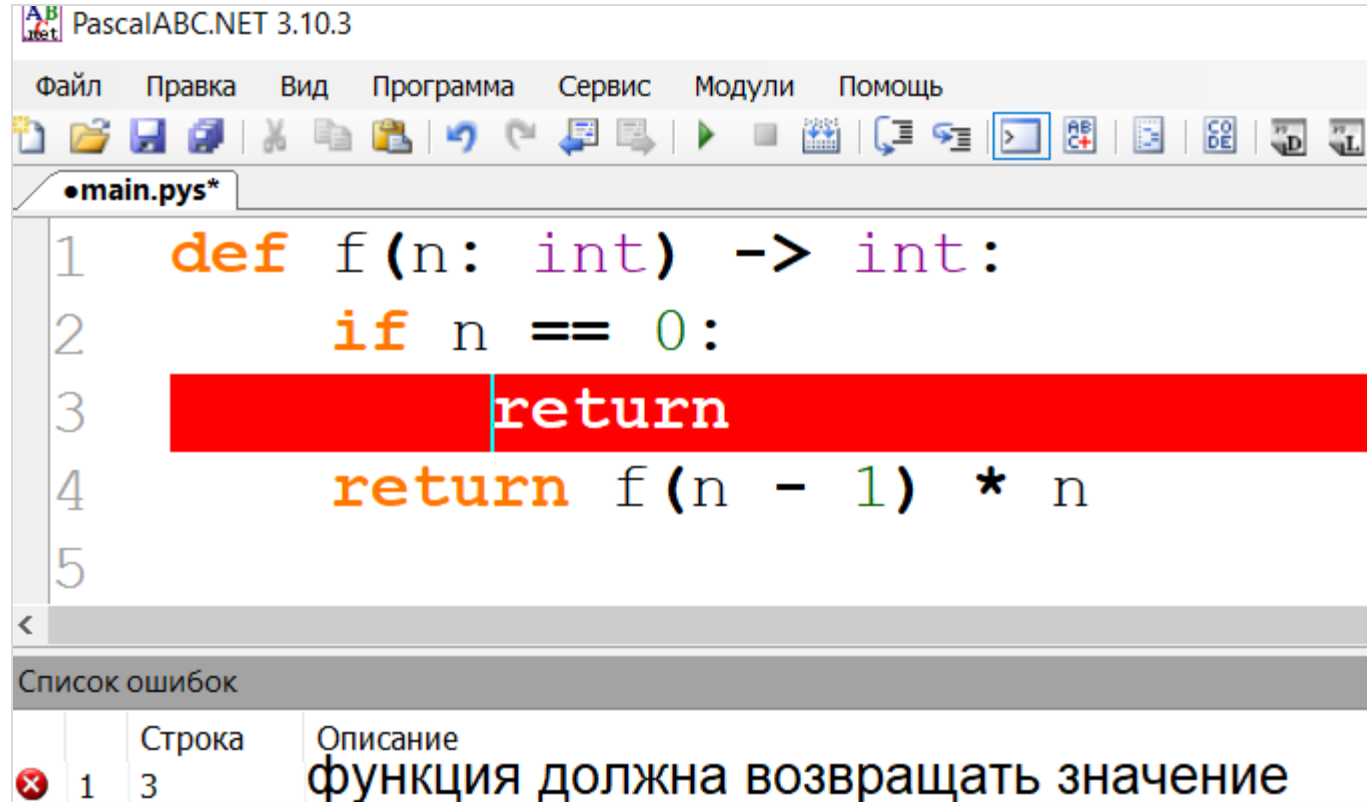
The screenshot shows the PascalABC.NET 3.10.3 IDE. The menu bar includes 'Файл', 'Правка', 'Вид', 'Программа', 'Сервис', 'Модули', and 'Помощь'. The toolbar contains various icons for file operations, editing, and execution. The active window is 'main.pys*'. The code editor contains the following Python code:

```
1 def f(a: int):  
2     print('123' + a)  
3  
4  
5 f(4)
```

The expression `'123' + a` on line 2 is highlighted in red, indicating a runtime error. Below the code editor is a 'Список ошибок' (Error List) panel. It contains one error entry:

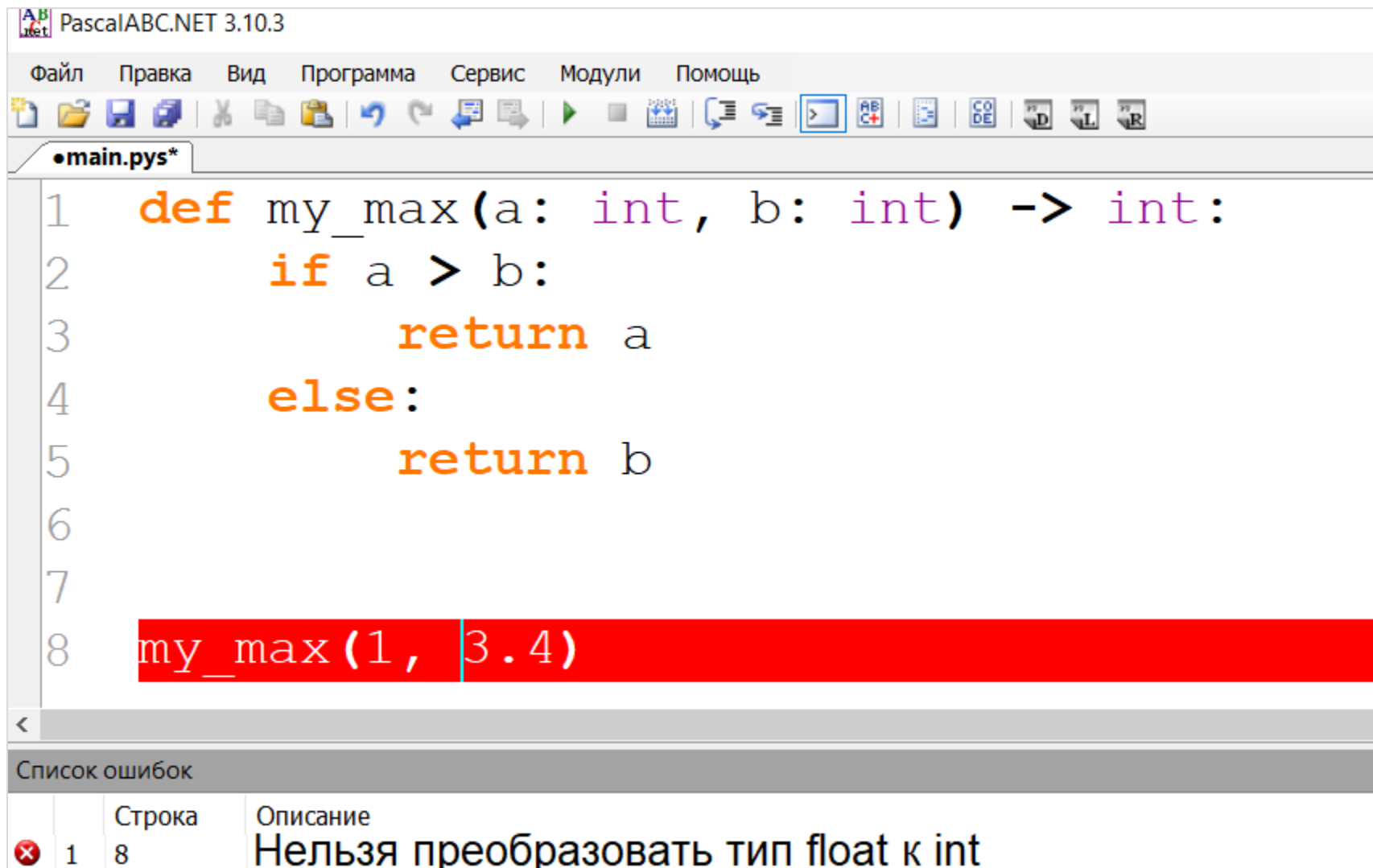
	Строка	Описание
✖ 1	2	Операция + неприменима к типам str и int

Функция не возвращает значение



```
PascalABC.NET 3.10.3
Файл  Правка  Вид  Программа  Сервис  Модули  Помощь
•main.pys*
1  def f(n: int) -> int:
2      if n == 0:
3          return
4      return f(n - 1) * n
5
Список ошибок
x 1  Строка  Описание
   3  функция должна возвращать значение
```

Несоответствие типа параметра функции



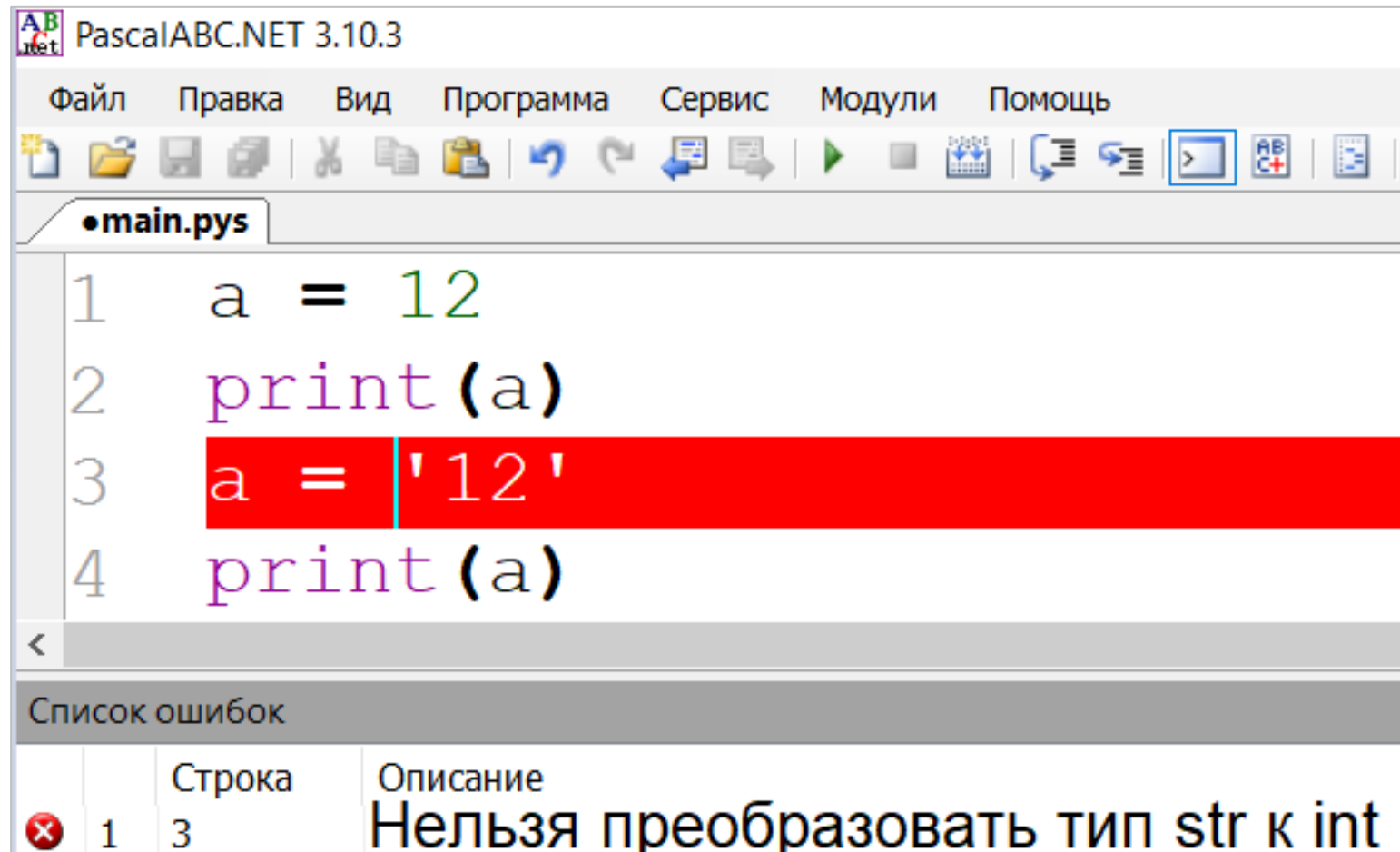
The screenshot shows the PascalABC.NET 3.10.3 IDE. The menu bar includes 'Файл', 'Правка', 'Вид', 'Программа', 'Сервис', 'Модули', and 'Помощь'. The toolbar contains various icons for file operations, editing, and execution. The main editor window displays a Python file named 'main.pys*' with the following code:

```
1 def my_max(a: int, b: int) -> int:
2     if a > b:
3         return a
4     else:
5         return b
6
7
8 my_max(1, 3.4)
```

The line `my_max(1, 3.4)` is highlighted in red. Below the editor, the 'Список ошибок' (Error List) panel shows a single error:

Строка	Описание
8	Нельзя преобразовать тип float к int

Несоответствие типа присваиваемого значения



The screenshot shows the PascalABC.NET 3.10.3 IDE. The menu bar includes 'Файл', 'Правка', 'Вид', 'Программа', 'Сервис', 'Модули', and 'Помощь'. The toolbar contains various icons for file operations, editing, and execution. The active file is 'main.pys'. The code editor contains the following Python code:

```
1 a = 12
2 print(a)
3 a = '12'
4 print(a)
```

The third line, `a = '12'`, is highlighted in red. Below the code editor is a 'Список ошибок' (Error List) panel. It contains one error entry:

	Строка	Описание
✖ 1	3	Нельзя преобразовать тип str к int

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

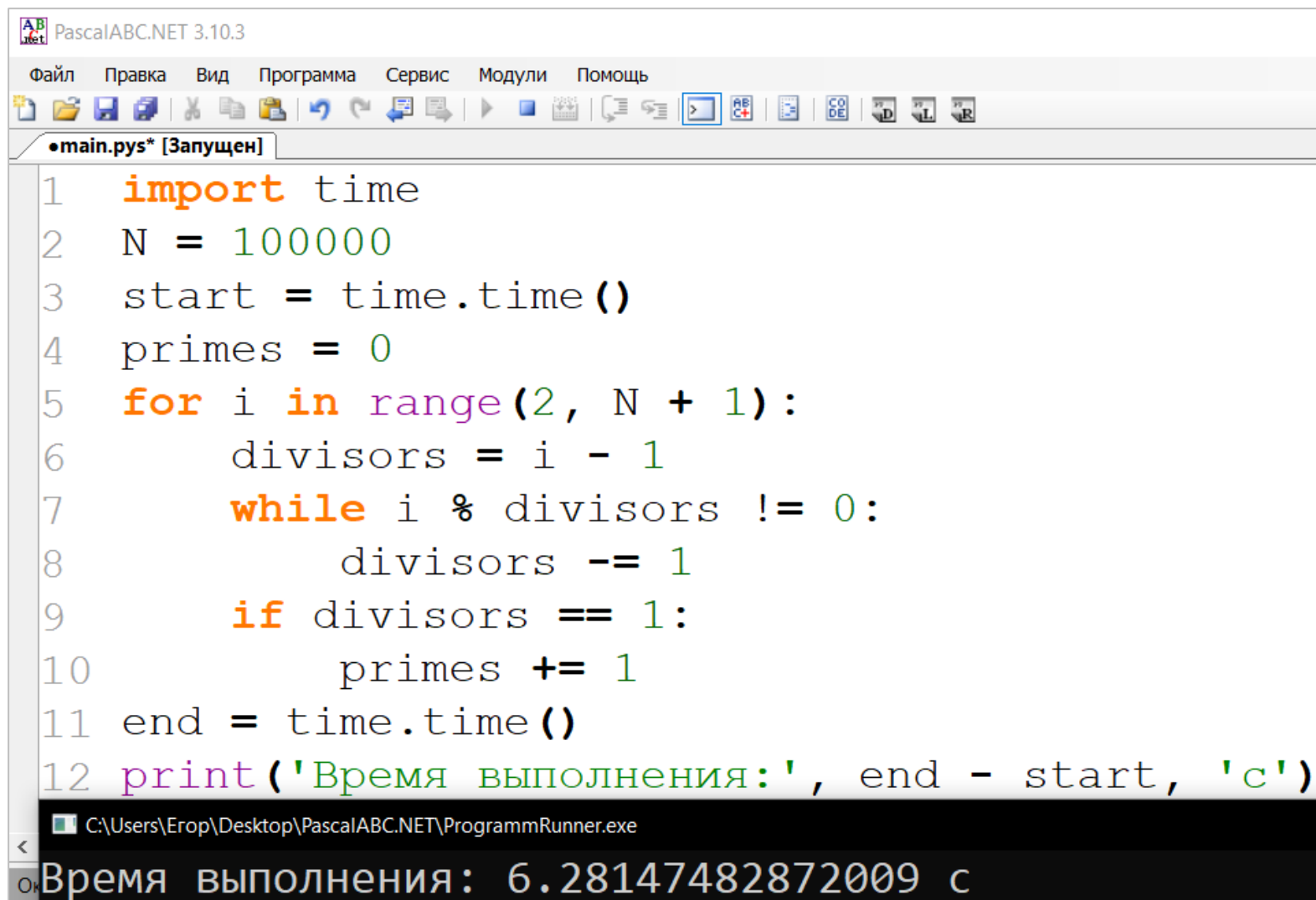
Тестирование времени работы программы

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



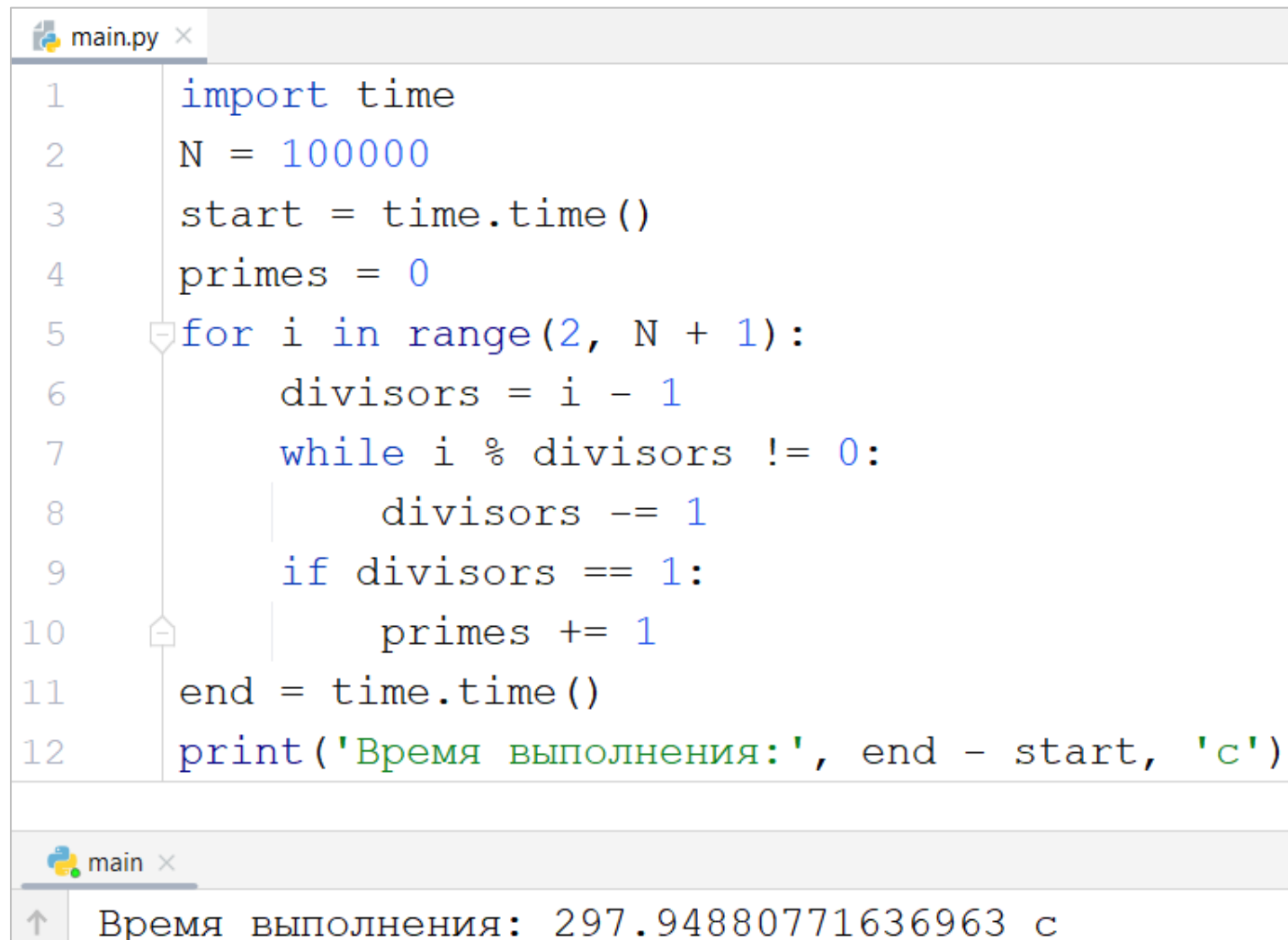
The screenshot shows the PascalABC.NET 3.10.3 IDE. The menu bar includes 'Файл', 'Правка', 'Вид', 'Программа', 'Сервис', 'Модули', and 'Помощь'. The toolbar contains various icons for file operations and execution. The active window is 'main.pys* [Запущен]'. The code in the editor is a Python script for finding prime numbers. The output window at the bottom shows the execution path and the execution time.

```
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, 'с')
```

C:\Users\Erop\Desktop\PascalABC.NET\ProgrammRunner.exe

Время выполнения: 6.28147482872009 с

Результаты замеров времени: Python

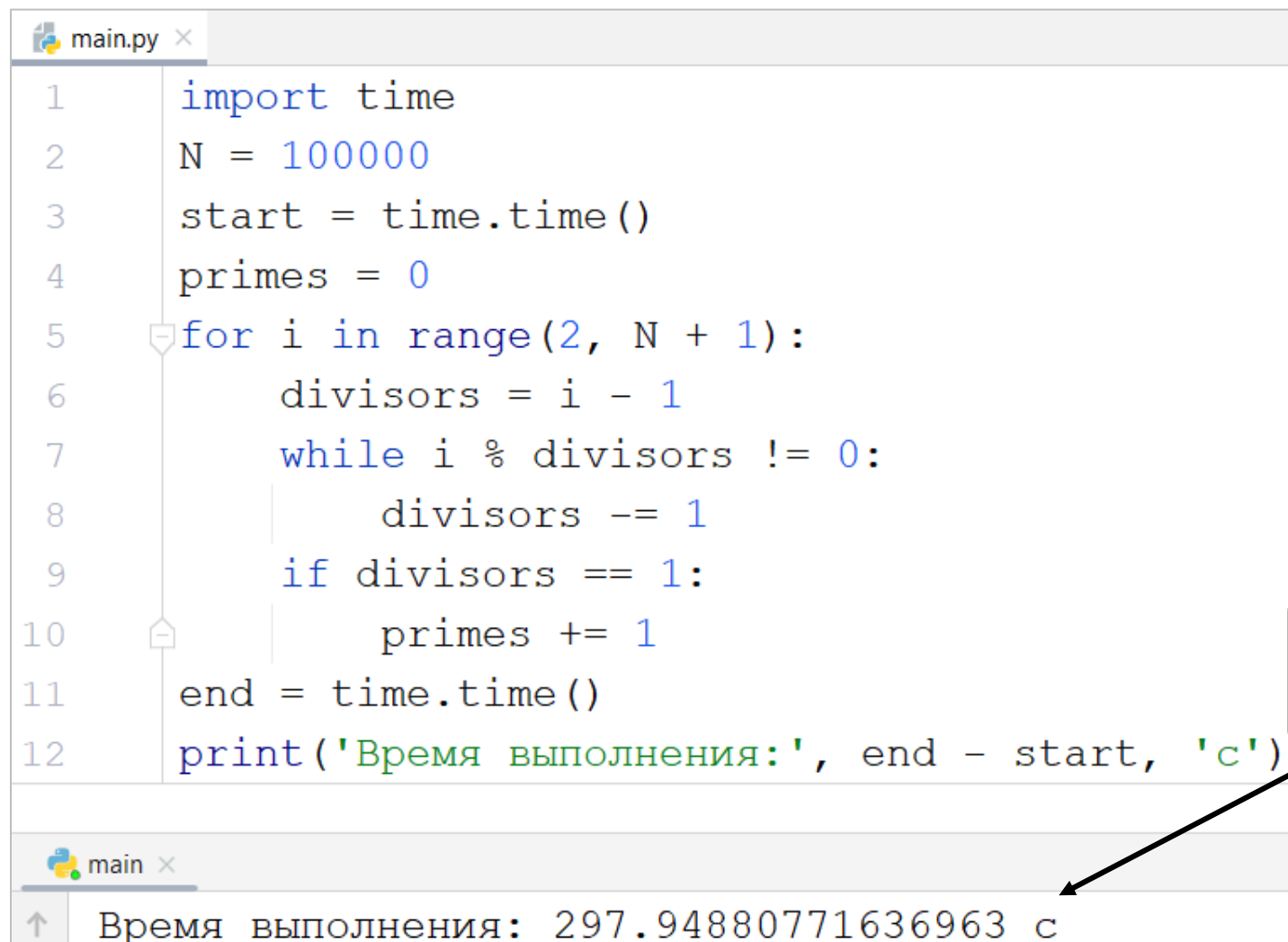


```
main.py x
1  import time
2  N = 100000
3  start = time.time()
4  primes = 0
5  for i in range(2, N + 1):
6      divisors = i - 1
7      while i % divisors != 0:
8          divisors -= 1
9      if divisors == 1:
10         primes += 1
11 end = time.time()
12 print('Время выполнения:', end - start,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

```
main x
↑  Время выполнения: 297.94880771636963 с
```

The image shows a Python IDE window titled 'main.py'. The code defines a function to count primes up to N=100,000 by checking for divisors. It uses the 'time' module to measure the execution time. The output at the bottom shows the execution took approximately 298 seconds.

Результаты замеров времени: Python



```
main.py x
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, 'с')
```

main x

↑ Время выполнения: 297.94880771636963 с

В 47 раз
медленнее!

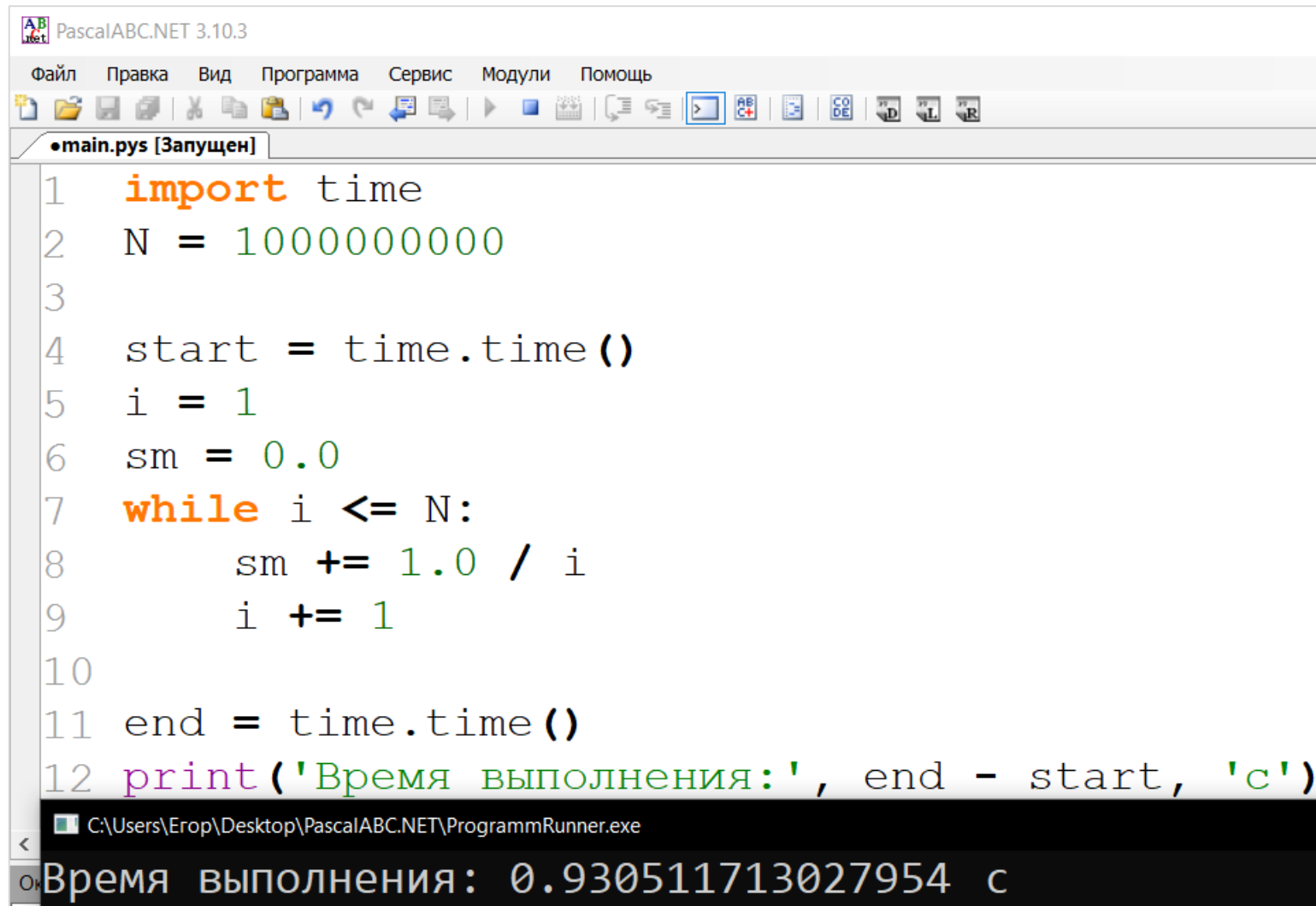
Тестирование времени работы программы

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

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

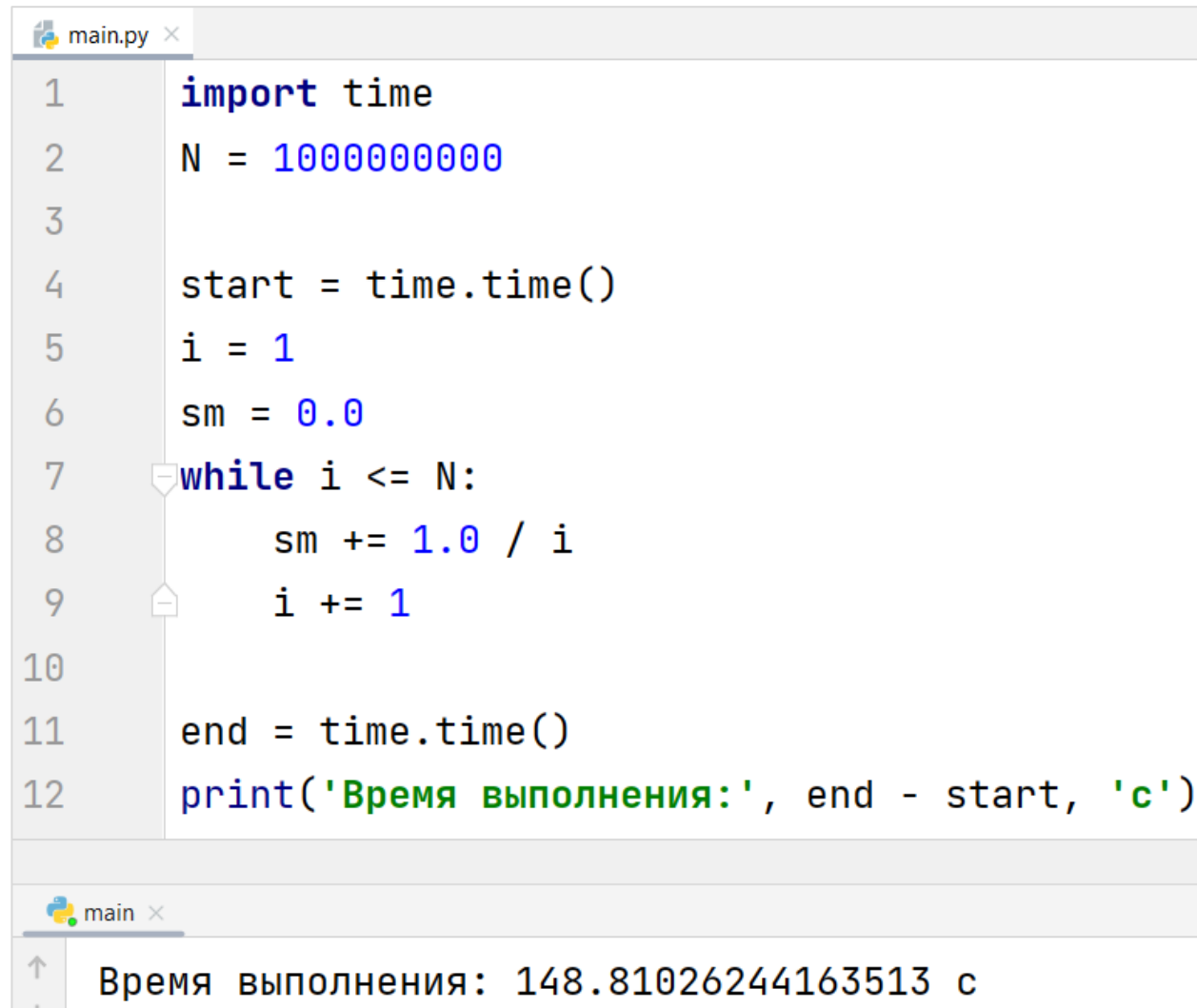
Для замеров использовался персональный компьютер с процессором Intel Core i7-8700K CPU 3.70GHz.

Результаты замеров времени: SPython



```
PascalABC.NET 3.10.3
Файл  Правка  Вид  Программа  Сервис  Модули  Помощь
[Icons]
•main.pys [Запущен]
1  import time
2  N = 10000000000
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, 'с')
C:\Users\Erop\Desktop\PascalABC.NET\ProgrammRunner.exe
Время выполнения: 0.930511713027954 с
```

Результаты замеров времени: Python



```
main.py x
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, 'с')
```

```
main x
↑  Время выполнения: 148.81026244163513 с
```

The image shows a screenshot of a Python IDE. The top pane, titled 'main.py', contains a script that imports the 'time' module, sets a large constant 'N' to 1,000,000,000, records the start time, and then enters a 'while' loop that iterates from 1 to 'N', calculating the sum of reciprocals (1/i). After the loop, it records the end time and prints the execution time in seconds. The bottom pane, titled 'main', shows the output of the script: 'Время выполнения: 148.81026244163513 с'.

Результаты замеров времени: Python

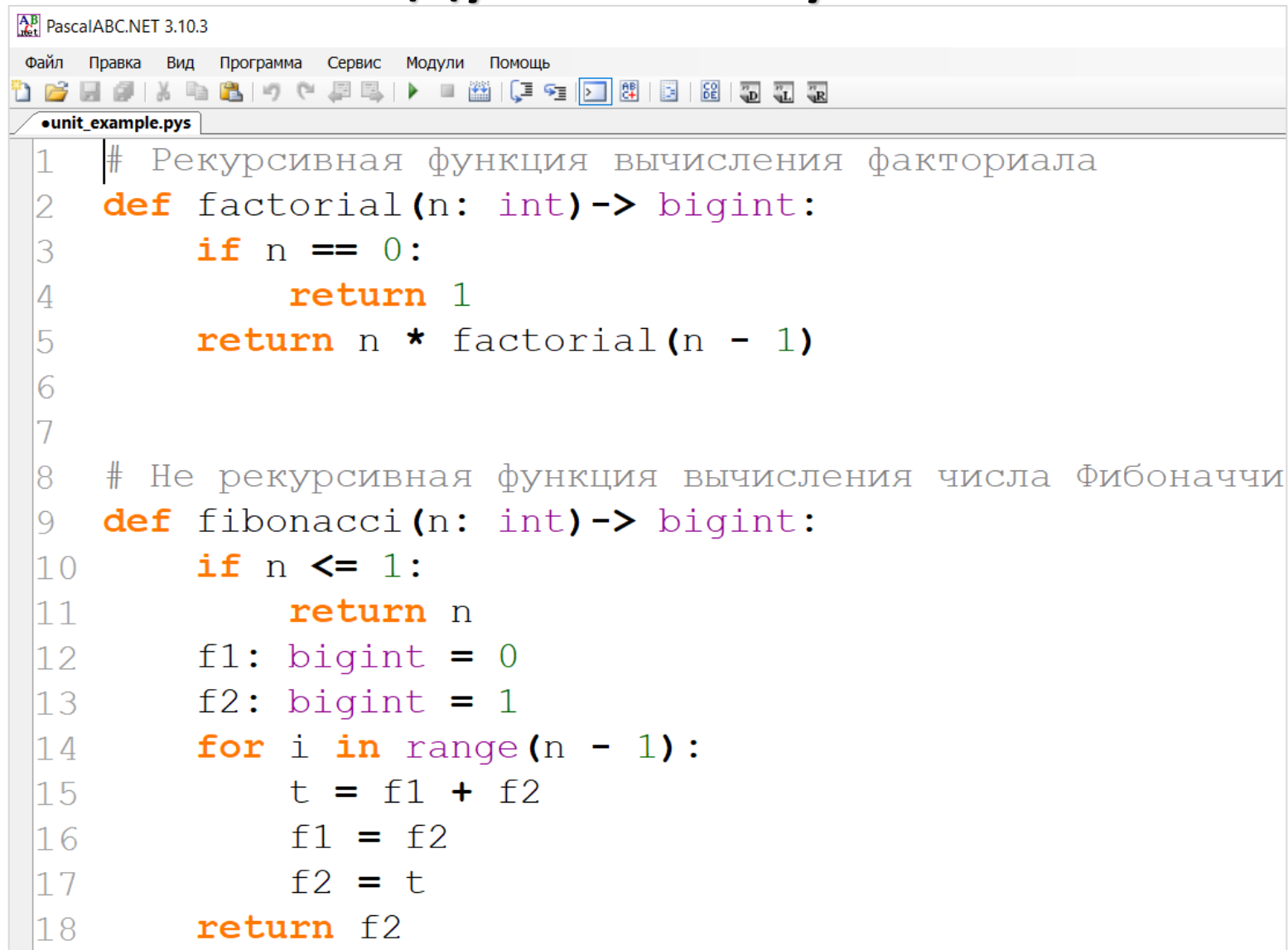
```
main.py x
1  import time
2  N = 1000000000
3
4  start = time.time()
5  i = 1
6  sm = 0.0
7  while i <= N:
8      sm += 1.0 / i
9      i += 1
10
11  end = time.time()
12  print('Время выполнения:', end - start, 'с')
```

```
main x
↑  Время выполнения: 148.81026244163513 с
```

В 160 раз
медленнее!

Взаимодействие SPython и PascalABC.NET

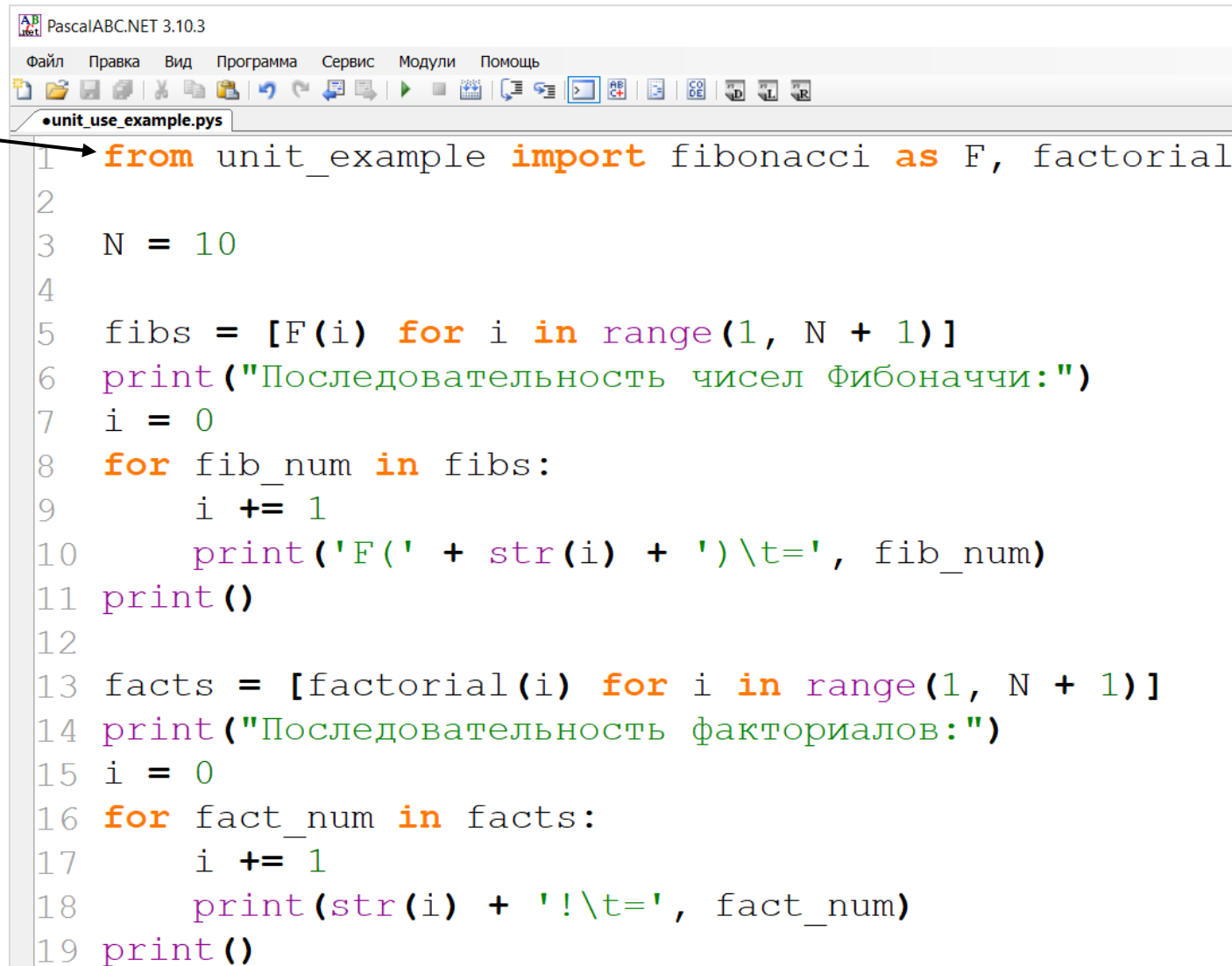
Модуль на SPython



```
1  # Рекурсивная функция вычисления факториала
2  def factorial(n: int)-> bigint:
3      if n == 0:
4          return 1
5      return n * factorial(n - 1)
6
7
8  # Не рекурсивная функция вычисления числа Фибоначчи
9  def fibonacci(n: int)-> bigint:
10     if n <= 1:
11         return n
12     f1: bigint = 0
13     f2: bigint = 1
14     for i in range(n - 1):
15         t = f1 + f2
16         f1 = f2
17         f2 = t
18     return f2
```

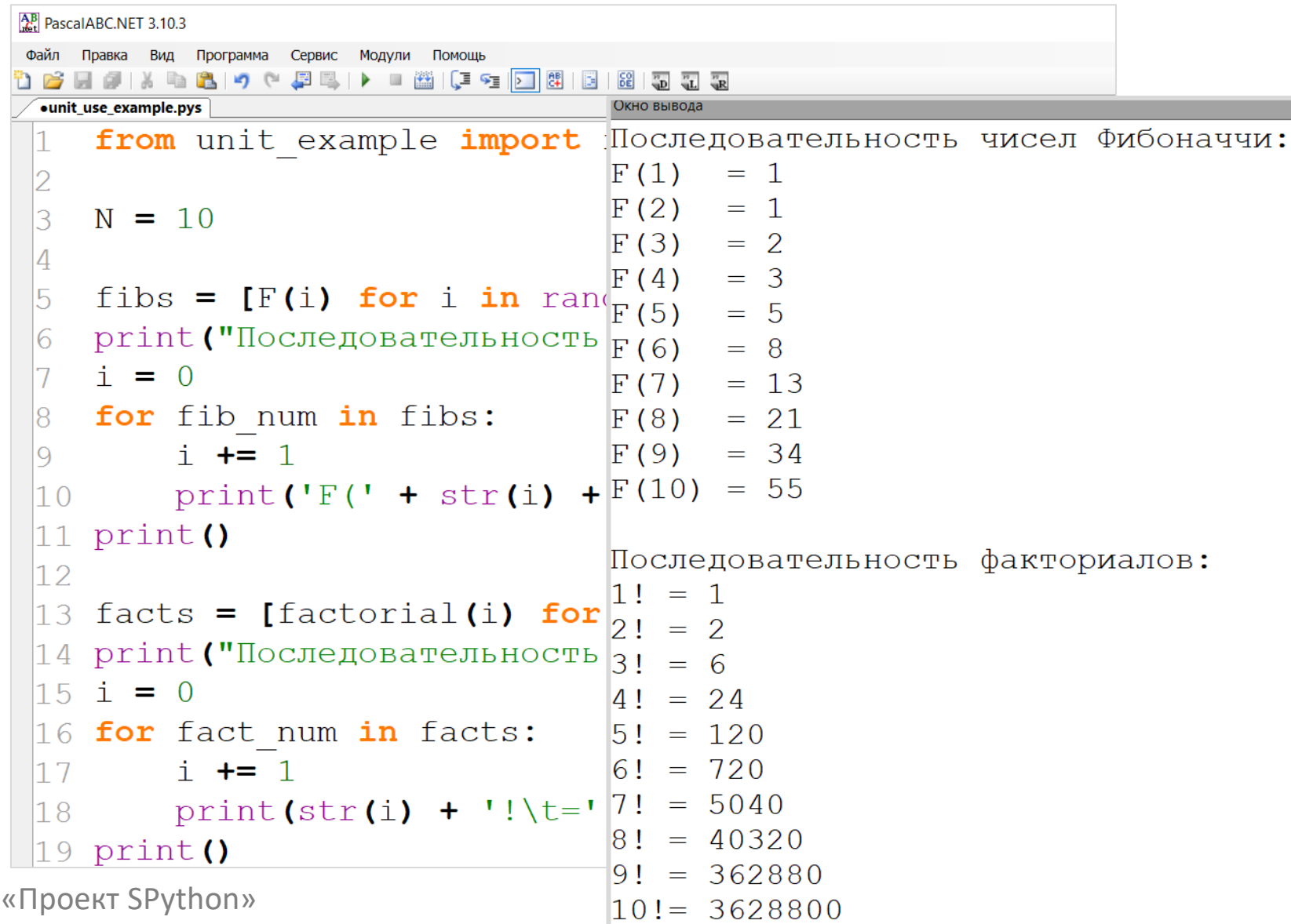
Использование модуля SPython на SPython

Подключение
модуля



```
1 from unit_example import fibonacci as F, factorial
2
3 N = 10
4
5 fibs = [F(i) for i in range(1, N + 1)]
6 print("Последовательность чисел Фибоначчи:")
7 i = 0
8 for fib_num in fibs:
9     i += 1
10    print('F(' + str(i) + ')\t=', fib_num)
11 print()
12
13 facts = [factorial(i) for i in range(1, N + 1)]
14 print("Последовательность факториалов:")
15 i = 0
16 for fact_num in facts:
17     i += 1
18    print(str(i) + '! \t=', fact_num)
19 print()
```

Использование модуля SPython на SPython



The screenshot shows the PascalABC.NET 3.10.3 IDE. The main editor window displays a Python script named `unit_use_example.pys`. The script imports a module `unit_example` and uses its `F` function to calculate Fibonacci numbers and its `factorial` function to calculate factorials. The output window on the right shows the results of these calculations.

```
1 from unit_example import F
2
3 N = 10
4
5 fibs = [F(i) for i in range(1, N+1)]
6 print("Последовательность чисел Фибоначчи:")
7 i = 0
8 for fib_num in fibs:
9     i += 1
10    print('F(' + str(i) + ') = ' + str(fib_num))
11 print()
12
13 facts = [factorial(i) for i in range(1, N+1)]
14 print("Последовательность факториалов:")
15 i = 0
16 for fact_num in facts:
17     i += 1
18    print(str(i) + '! = ' + str(fact_num))
19 print()
```

Окно вывода

Последовательность чисел Фибоначчи:

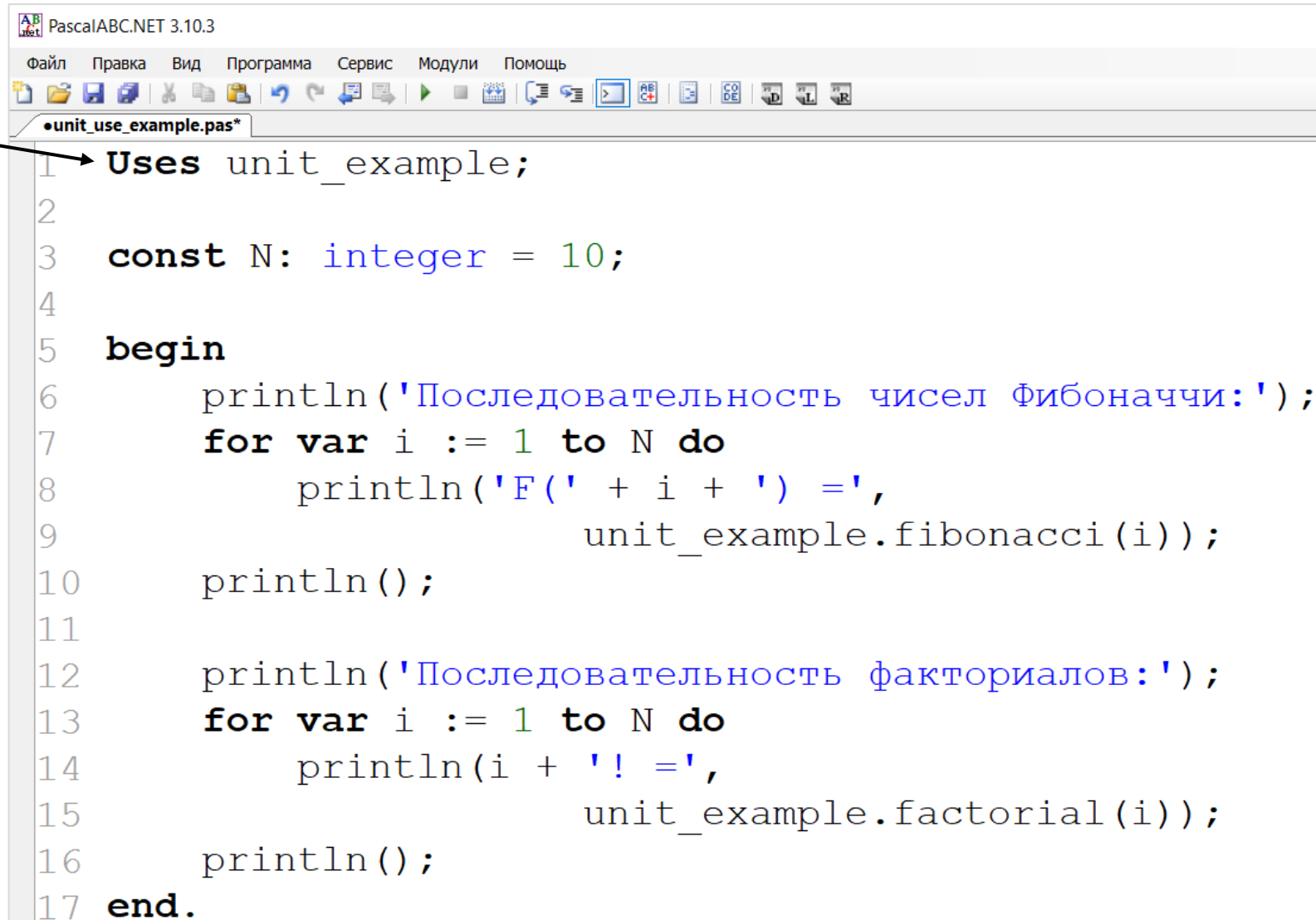
```
F(1) = 1
F(2) = 1
F(3) = 2
F(4) = 3
F(5) = 5
F(6) = 8
F(7) = 13
F(8) = 21
F(9) = 34
F(10) = 55
```

Последовательность факториалов:

```
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
```

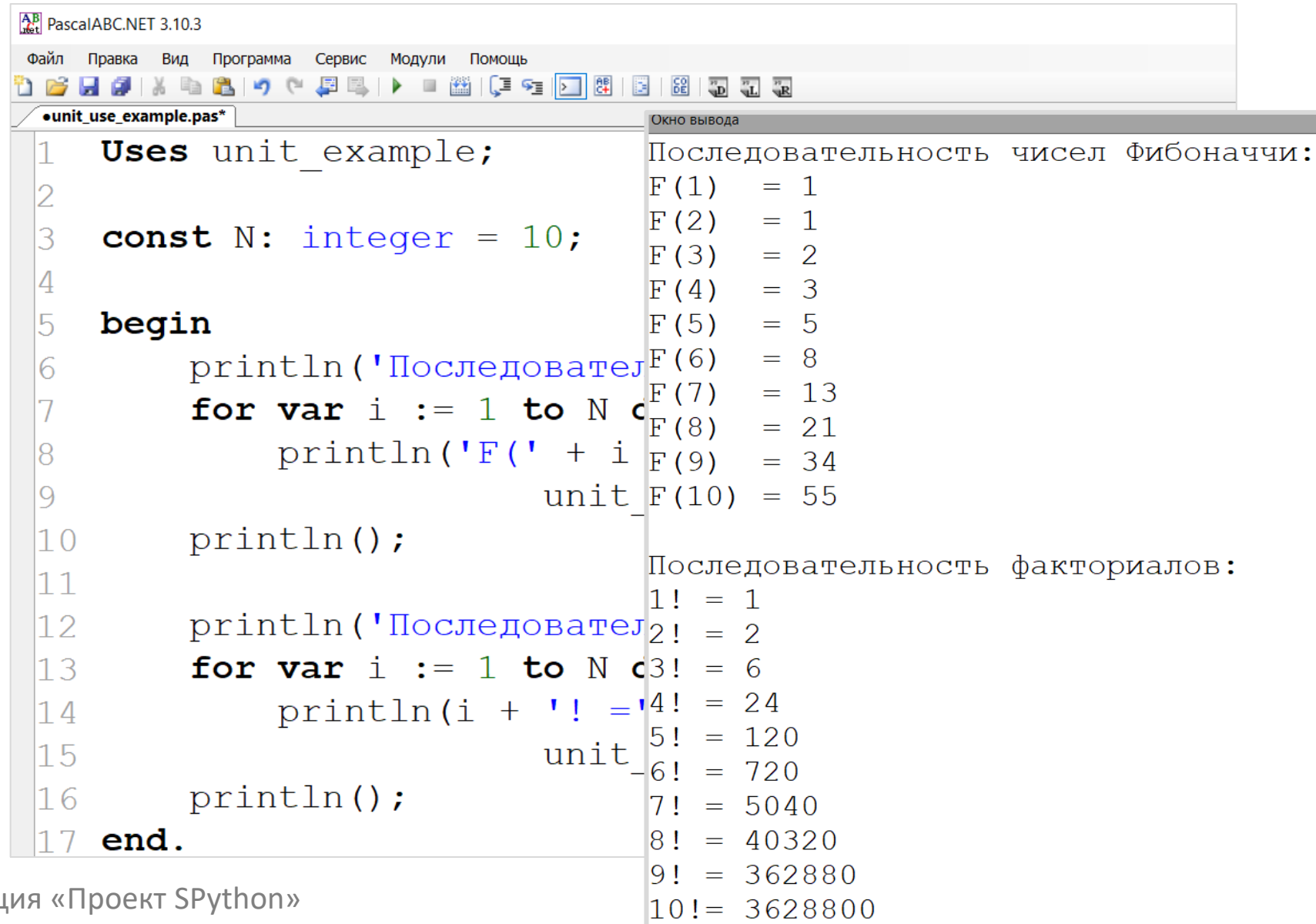
Использование модуля SPython на PascalABC.NET

Подключение
модуля



```
PascalABC.NET 3.10.3
Файл  Правка  Вид  Программа  Сервис  Модули  Помощь
unit_use_example.pas*
1  Uses unit_example;
2
3  const N: integer = 10;
4
5  begin
6      println('Последовательность чисел Фибоначчи:');
7      for var i := 1 to N do
8          println('F(' + i + ') =',
9                  unit_example.fibonacci(i));
10     println();
11
12     println('Последовательность факториалов:');
13     for var i := 1 to N do
14         println(i + '! =',
15                 unit_example.factorial(i));
16     println();
17 end.
```

Использование модуля SPython на PascalABC.NET



The screenshot shows the PascalABC.NET 3.10.3 IDE. The main editor window displays a Pascal program named `unit_use_example.pas`. The program uses the `SPython` module to calculate and print the Fibonacci sequence and the factorial sequence. The output window on the right shows the results of these calculations.

```
1  Uses unit_example;  
2  
3  const N: integer = 10;  
4  
5  begin  
6      println('Последовательность чисел Фибоначчи:');  
7      for var i := 1 to N do  
8          println('F(' + i + ') = ' + SPython.Fibonacci(i));  
9      println();  
10     println('Последовательность факториалов:');  
11     for var i := 1 to N do  
12         println(i + '! = ' + SPython.Factorial(i));  
13     println();  
14 end.
```

Окно вывода

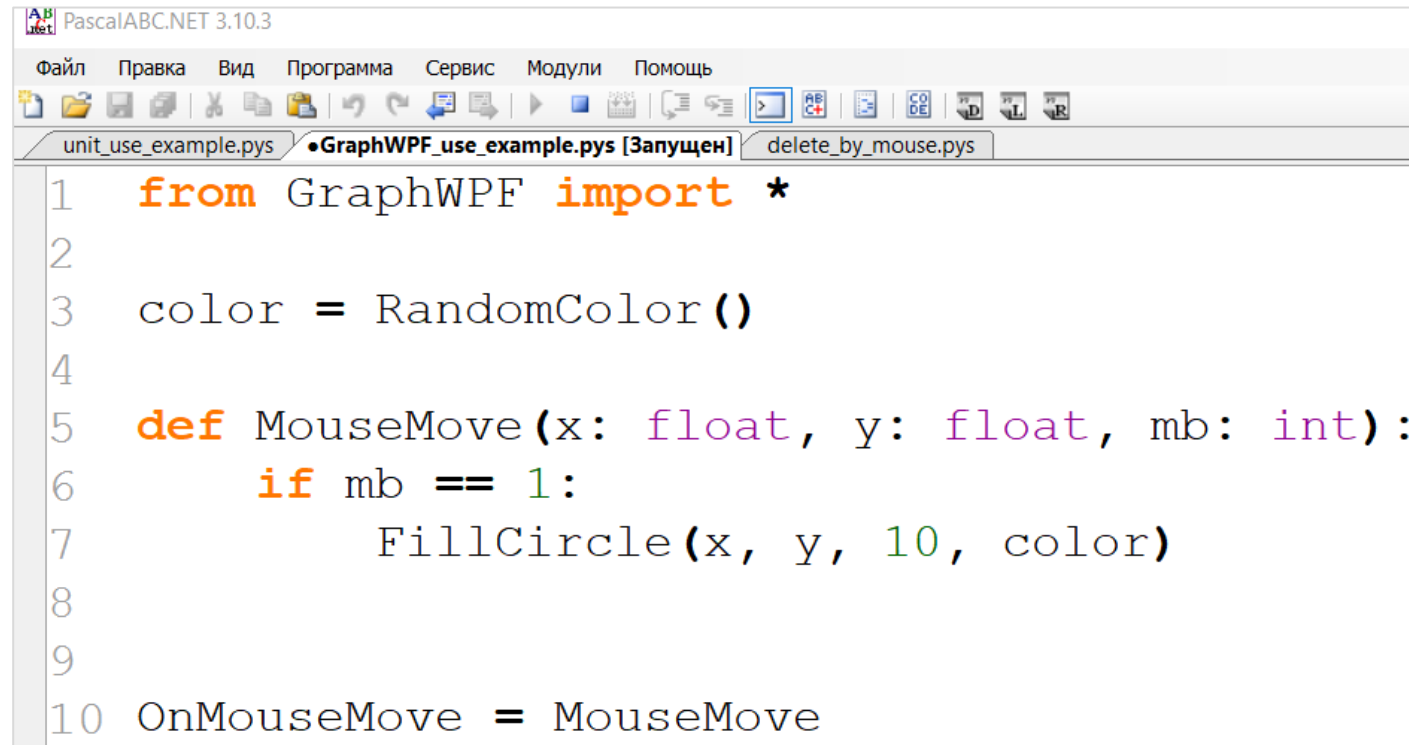
Последовательность чисел Фибоначчи:

F(1)	=	1
F(2)	=	1
F(3)	=	2
F(4)	=	3
F(5)	=	5
F(6)	=	8
F(7)	=	13
F(8)	=	21
F(9)	=	34
F(10)	=	55

Последовательность факториалов:

1!	=	1
2!	=	2
3!	=	6
4!	=	24
5!	=	120
6!	=	720
7!	=	5040
8!	=	40320
9!	=	362880
10!	=	3628800

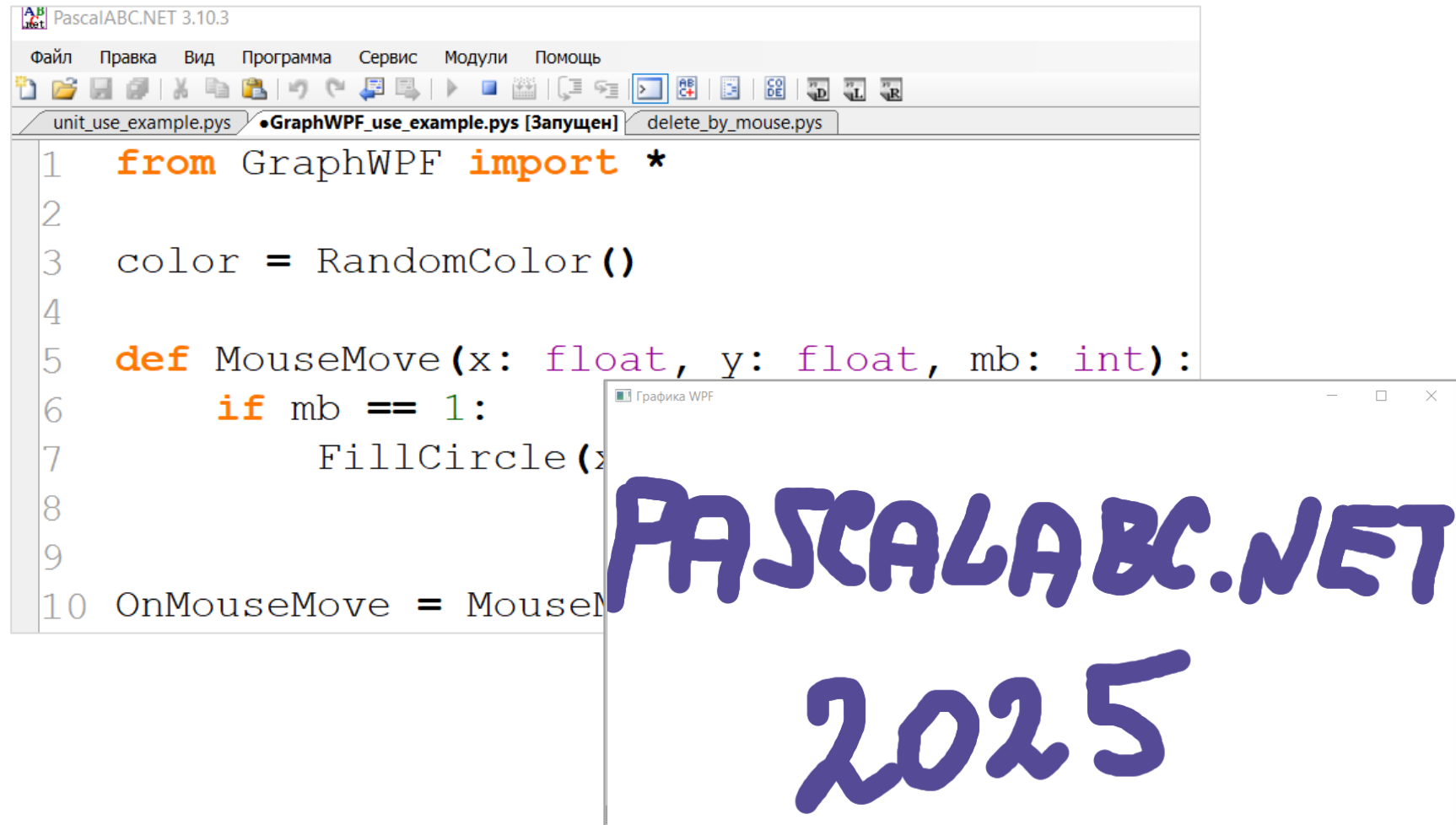
Использование модуля GraphWPF на SPython



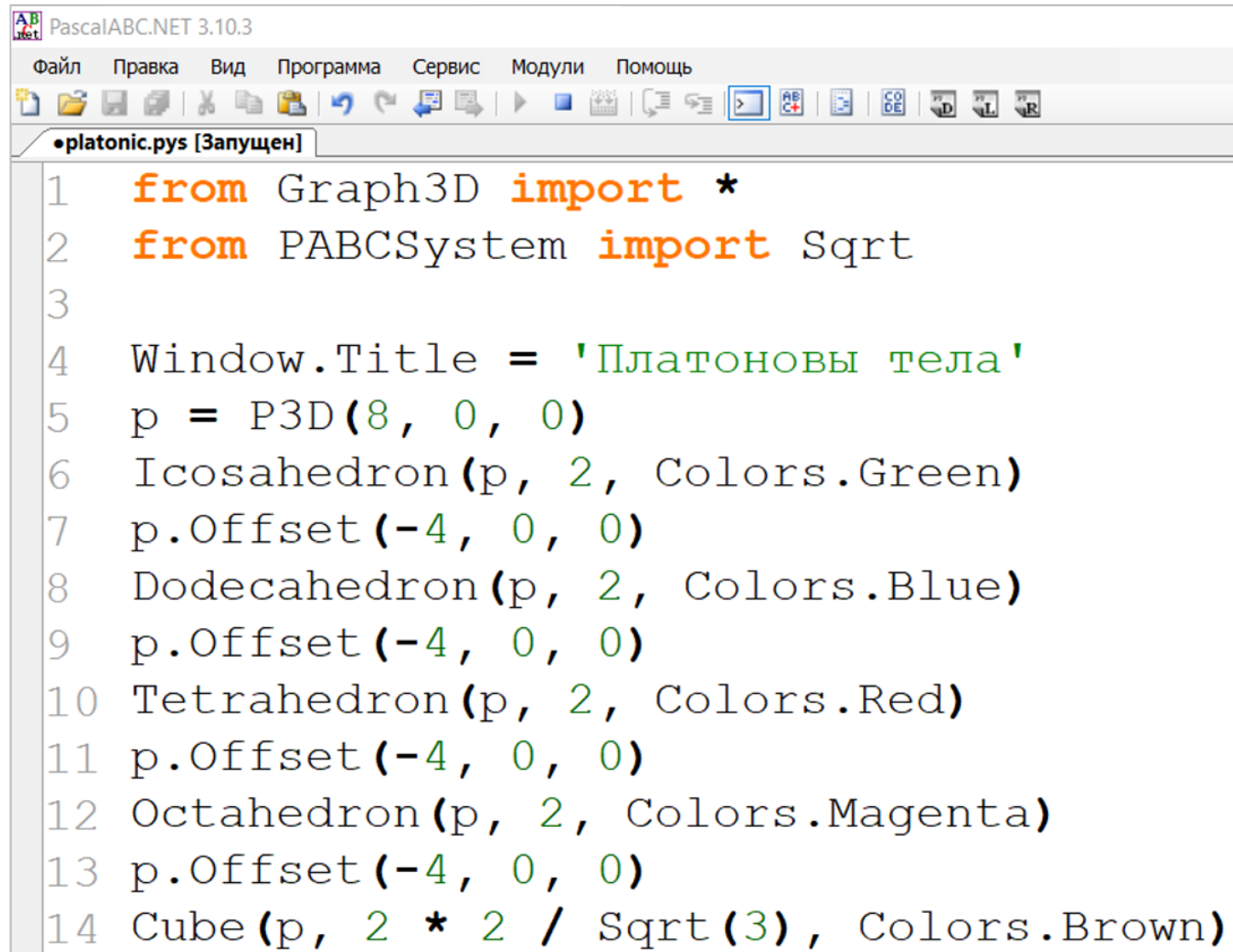
The screenshot shows the PascalABC.NET 3.10.3 IDE. The menu bar includes 'Файл', 'Правка', 'Вид', 'Программа', 'Сервис', 'Модули', and 'Помощь'. The toolbar contains various icons for file operations and execution. The tab bar shows three files: 'unit_use_example.pys', '•GraphWPF_use_example.pys [Запущен]', and 'delete_by_mouse.pys'. The main editor window displays the following Python code:

```
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, y, 10, color)
8
9
10 OnMouseMove = MouseMove
```

Использование модуля GraphWPF на SPython



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 = '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

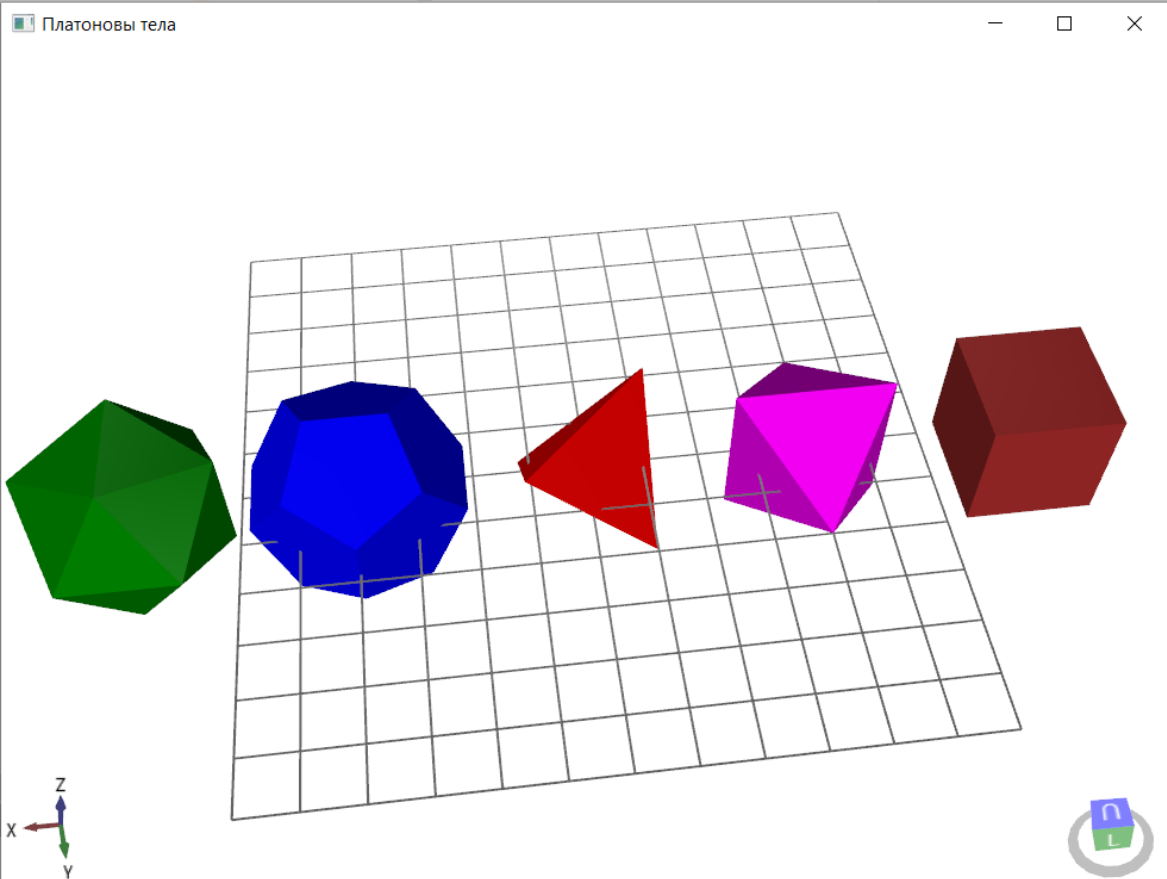
PascalABC.NET 3.10.3

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(p
7 p.Offset(-4,
8 Dodecahedron
9 p.Offset(-4,
10 Tetrahedron(p
11 p.Offset(-4,
12 Octahedron(p
13 p.Offset(-4,
14 Cube(p, 2 * 2
```

Платоновы тела



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

