

# Программирование. Язык Python.

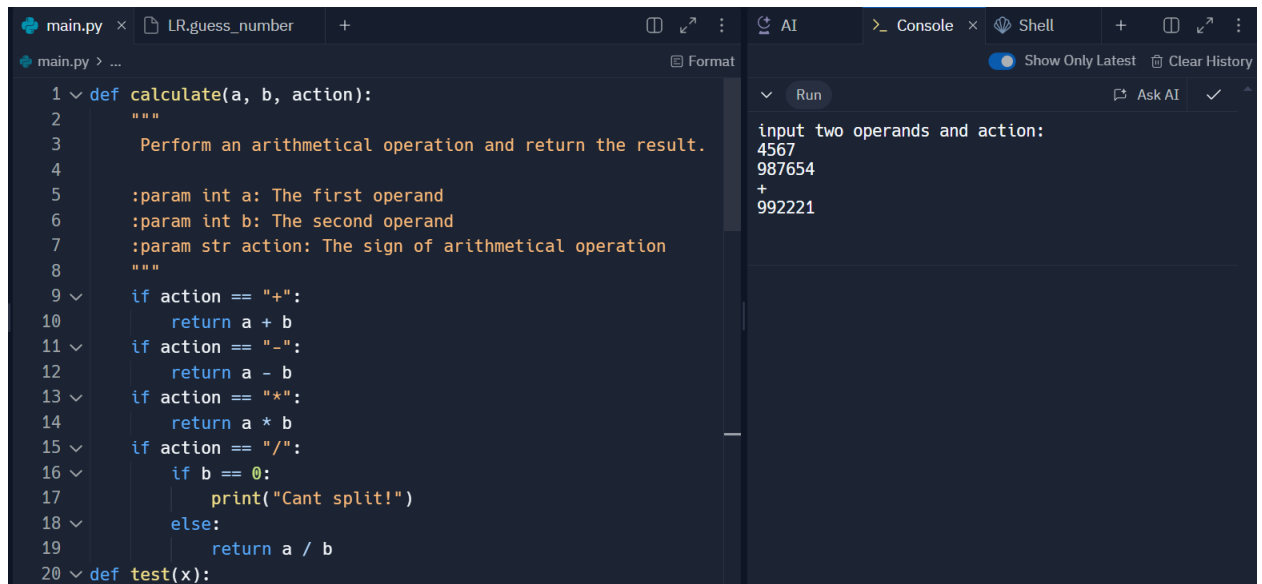
## Лабораторная работа № 1.

Ссылка на replit: <https://replit.com/@Olljux/Herzenlabs#main.py>

Ссылка на Git репозиторий: [https://github.com/Olljux/Herzen\\_proga\\_2semester](https://github.com/Olljux/Herzen_proga_2semester)

**3.1:** Создайте простую программу калькулятор, которая позволяет из функции `main()` ввести два числа и тип арифметической операции, а потом вычисляет результат. Свой код опубликуйте на KttSsreSlit. cRm и предоставьте ссылку в ответах на лабораторную работу в Moodle в документе-отчёте. Реализацию арифметических действий и вычисление результата с его возвратом сделайте в отдельной функции `calculate(...)`. Протестируйте свой калькулятор с помощью вызова нескольких своих простых функций `test_*`() с ключевым словом `assert` внутри. Обязательно напишите хорошую документацию к своему коду.

Результат работы кода:



```
1 def calculate(a, b, action):
2     """
3     Perform an arithmetical operation and return the result.
4
5     :param int a: The first operand
6     :param int b: The second operand
7     :param str action: The sign of arithmetical operation
8     """
9     if action == "+":
10         return a + b
11     if action == "-":
12         return a - b
13     if action == "*":
14         return a * b
15     if action == "/":
16         if b == 0:
17             print("Cant split!")
18         else:
19             return a / b
20 def test(x):
```

Run

input two operands and action:  
4567  
987654  
+  
992221

```
main.py x LR.guess_number +
main.py > ...
21 """
22 Testing the function calculate.
23
24 :param str x: The sign of arithmetical operation
25 """
26 if x == "+":
27     assert calculate(5, 7, "+") == 12 #True
28 if x == "-":
29     assert calculate(6, 7, "-") == 99 #False
30 if x == "*":
31     assert calculate(5, 7, "*") == 35 #True
32 if x == "/":
33     assert calculate(10, 2, "/") == 7 #False
34 def main():
35     """
36     Input of operands and operation, compute the result using
37     the function calculate.
38     """
39     print("input two operands and action:")
40     a = int(input())
```

Console: Run

```
input two operands and action:
456
876
*
399456
```

```
main.py x LR.guess_number +
main.py > ...
40 b = int(input())
41 action = str(input())
42 print(calculate(a, b, action))
43 test(action)
44
45 main()
```

Console: Run

```
input two operands and action:
456
0
/
Cant split!
None
Traceback (most recent call last):
  File "/home/runner/Herzenlabs/main.py", line 4
    5, in <module>
      main()
      File "/home/runner/Herzenlabs/main.py", line 4
    3, in main
      test(action)
      File "/home/runner/Herzenlabs/main.py", line 3
    3, in test
      assert calculate(10, 2, "/") == 7 #False
      ~~~~~
AssertionError
```

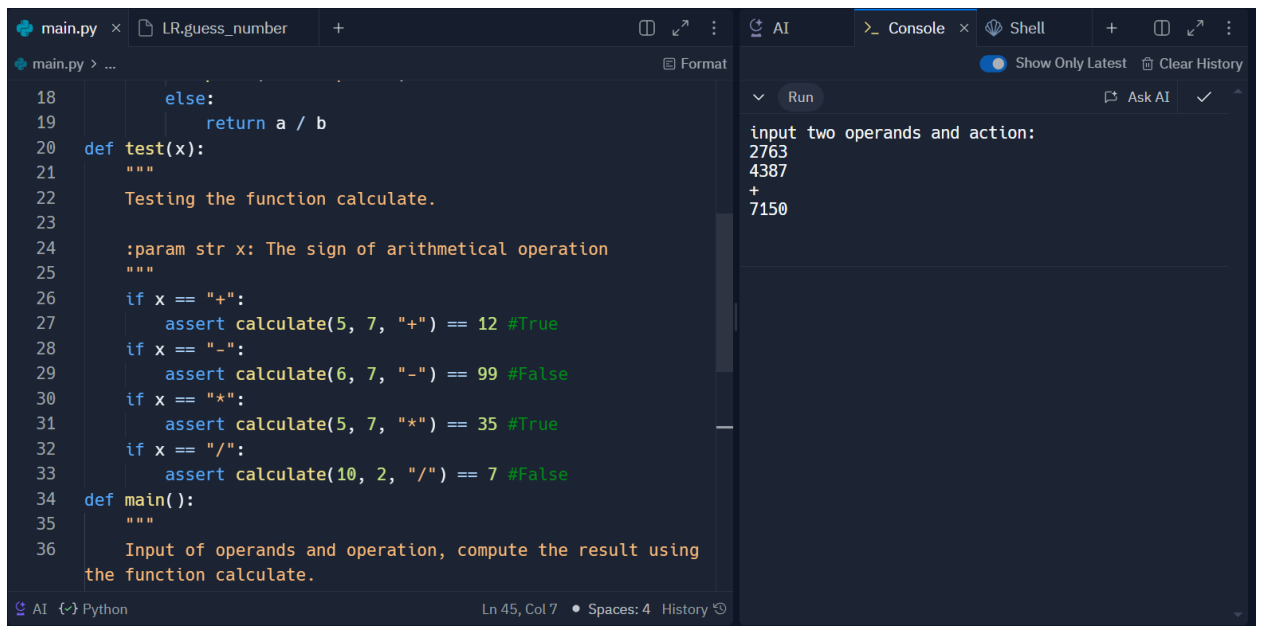
## Тест с помощью assert:

```
main.py x LR.guess_number +
main.py > ...
18 else:
19     return a / b
20 def test(x):
21     """
22     Testing the function calculate.
23
24     :param str x: The sign of arithmetical operation
25     """
26     if x == "+":
27         assert calculate(5, 7, "+") == 12 #True
28     if x == "-":
29         assert calculate(6, 7, "-") == 99 #False
30     if x == "*":
31         assert calculate(5, 7, "*") == 35 #True
32     if x == "/":
33         assert calculate(10, 2, "/") == 7 #False
34 def main():
35     """
36     Input of operands and operation, compute the result using
37     the function calculate.
38     """
39     print("input two operands and action:")
40     a = int(input())
41     b = int(input())
42     action = str(input())
43     print(calculate(a, b, action))
44     test(action)
45     main()
```

Console: Run

```
input two operands and action:
1234567
8765
-
1225802
Traceback (most recent call last):
  File "/home/runner/Herzenlabs/main.py", line 4
    5, in <module>
      main()
      File "/home/runner/Herzenlabs/main.py", line 4
    3, in main
      test(action)
      File "/home/runner/Herzenlabs/main.py", line 2
    9, in test
      assert calculate(6, 7, "-") == 99 #False
      ~~~~~
AssertionError
```

Ln 45, Col 7 • Spaces: 4 History



The screenshot shows a code editor with a Python file named `main.py`. The code defines a `calculate` function and a `test` function. The `test` function contains several assertions. The console output shows the result of running the code, displaying the input operands and the action performed.

```
18         else:
19             return a / b
20 def test(x):
21     """
22     Testing the function calculate.
23
24     :param str x: The sign of arithmetical operation
25     """
26     if x == "+":
27         assert calculate(5, 7, "+") == 12 #True
28     if x == "-":
29         assert calculate(6, 7, "-") == 99 #False
30     if x == "*":
31         assert calculate(5, 7, "*") == 35 #True
32     if x == "/":
33         assert calculate(10, 2, "/") == 7 #False
34 def main():
35     """
36     Input of operands and operation, compute the result using
37     the function calculate.
```

Console output:

```
input two operands and action:
2763
4387
+
7150
```

Листинг:

```
def calculate(a, b, action):
```

```
    """
```

```
    Perform an arithmetical operation and return the result.
```

```
    :param int a: The first operand
```

```
    :param int b: The second operand
```

```
    :param str action: The sign of arithmetical operation
```

```
    """
```

```
    if action == "+":
```

```
        return a + b
```

```
    if action == "-":
```

```
        return a - b
```

```
    if action == "*":
```

```
        return a * b
```

```
    if action == "/":
```

```
        if b == 0:
```

```
            print("Cant split!")
```

```

        else:
            return a / b

def test(x):
    """
    Testing the function calculate.

    :param str x: The sign of arithmetical operation
    """
    if x == "+":
        assert calculate(5, 7, "+") == 12 #True
    if x == "-":
        assert calculate(6, 7, "-") == 99 #False
    if x == "*":
        assert calculate(5, 7, "*") == 35 #True
    if x == "/":
        assert calculate(10, 2, "/") == 7 #False

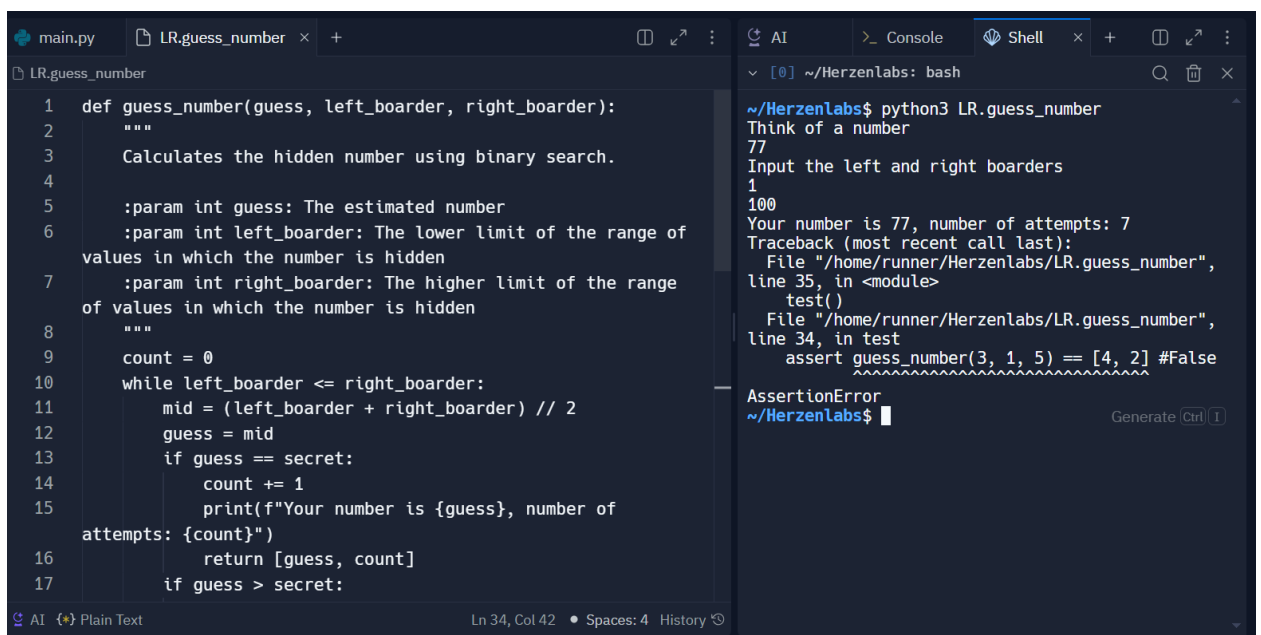
def main():
    """
    Input of operands and operation, compute the result using the function calculate.
    """
    print("input two operands and action:")
    a = int(input())
    b = int(input())
    action = str(input())
    print(calculate(a, b, action))
    test(action)

main()

```

**3.2:** Реализуйте программно классическую простую игру "угадай число" (guess number) с помощью алгоритма медленного перебора (инкремента) по одному числу, либо с помощью алгоритма бинарного поиска. Алгоритм принимает на вход само число, которое он должен угадать, интервал значений в котором оно загадано и в цикле делает угадывания тем или иным выбранным вами способом. После угадывания из функции алгоритма возвращается угаданное число и число угадываний/сравнений, которые пришлось проделать. Обязательно напишите хорошую документацию к своему коду.

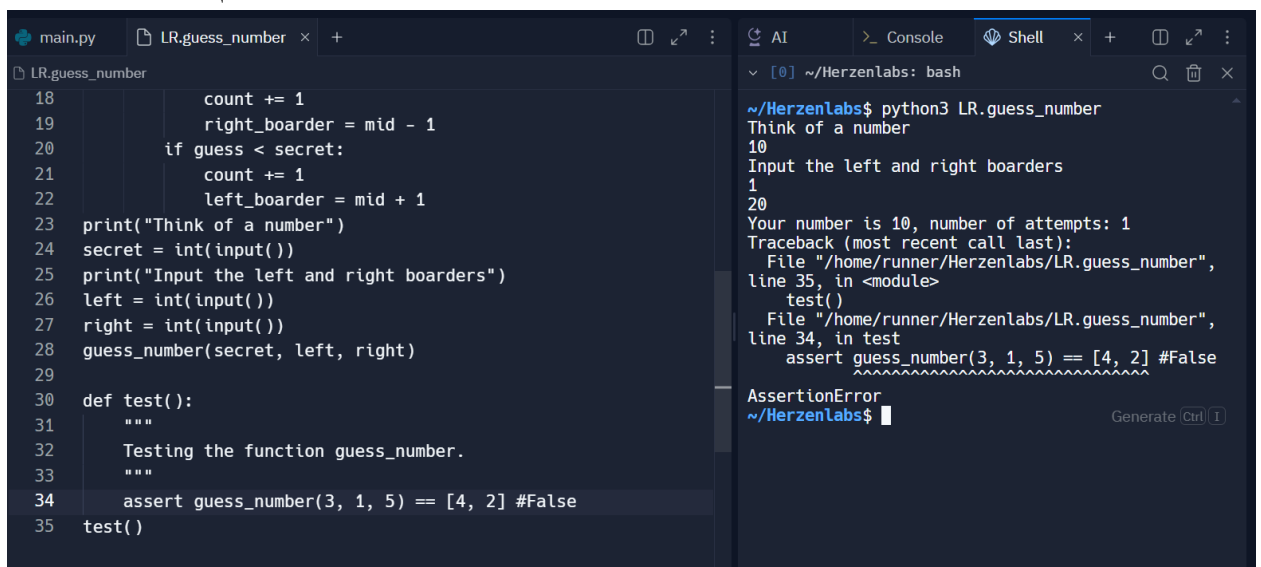
Результат работы кода:



```
main.py LR.guess_number x +
LR.guess_number
1 def guess_number(guess, left_boarder, right_boarder):
2     """
3     Calculates the hidden number using binary search.
4
5     :param int guess: The estimated number
6     :param int left_boarder: The lower limit of the range of
  values in which the number is hidden
7     :param int right_boarder: The higher limit of the range
  of values in which the number is hidden
8     """
9     count = 0
10    while left_boarder <= right_boarder:
11        mid = (left_boarder + right_boarder) // 2
12        guess = mid
13        if guess == secret:
14            count += 1
15            print(f"Your number is {guess}, number of
  attempts: {count}")
16            return [guess, count]
17        if guess > secret:
```

```
~/Herzenlabs$ python3 LR.guess_number
Think of a number
77
Input the left and right boarders
1
100
Your number is 77, number of attempts: 7
Traceback (most recent call last):
  File "/home/runner/Herzenlabs/LR.guess_number",
    line 35, in <module>
    test()
  File "/home/runner/Herzenlabs/LR.guess_number",
    line 34, in test
    assert guess_number(3, 1, 5) == [4, 2] #False
AssertionError
~/Herzenlabs$
```

Тест с помощью assert:



```
main.py LR.guess_number x +
LR.guess_number
18        count += 1
19        right_boarder = mid - 1
20        if guess < secret:
21            count += 1
22            left_boarder = mid + 1
23    print("Think of a number")
24    secret = int(input())
25    print("Input the left and right boarders")
26    left = int(input())
27    right = int(input())
28    guess_number(secret, left, right)
29
30    def test():
31        """
32        Testing the function guess_number.
33        """
34        assert guess_number(3, 1, 5) == [4, 2] #False
35    test()
```

```
~/Herzenlabs$ python3 LR.guess_number
Think of a number
10
Input the left and right boarders
1
20
Your number is 10, number of attempts: 1
Traceback (most recent call last):
  File "/home/runner/Herzenlabs/LR.guess_number",
    line 35, in <module>
    test()
  File "/home/runner/Herzenlabs/LR.guess_number",
    line 34, in test
    assert guess_number(3, 1, 5) == [4, 2] #False
AssertionError
~/Herzenlabs$
```

Листинг:

```
def guess_number(guess, left_boarder, right_boarder):
```

```
    """
```

```
        Calculates the hidden number using binary search.
```

```
        :param int guess: The estimated number
```

```
        :param int left_boarder: The lower limit of the range of values in which the  
number is hidden
```

```
        :param int right_boarder: The higher limit of the range of values in which the  
number is hidden
```

```
    """
```

```
    count = 0
```

```
    while left_boarder <= right_boarder:
```

```
        mid = (left_boarder + right_boarder) // 2
```

```
        guess = mid
```

```
        if guess == secret:
```

```
            count += 1
```

```
            print(f"Your number is {guess}, number of attempts: {count}")
```

```
            return [guess, count]
```

```
        if guess > secret:
```

```
            count += 1
```

```
            right_boarder = mid - 1
```

```
        if guess < secret:
```

```
            count += 1
```

```
            left_boarder = mid + 1
```

```
print("Think of a number")
```

```
secret = int(input())
```

```
print("Input the left and right boarders")
```

```
left = int(input())
```

```
right = int(input())
```

```
guess_number(secret, left, right)
```

```
def test():
```

```
    """
```

```
    Testing the function guess_number.
```

```
    """
```

```
    assert guess_number(3, 1, 5) == [4, 2] #False
```

```
test()
```