Программирование. Язык Python.

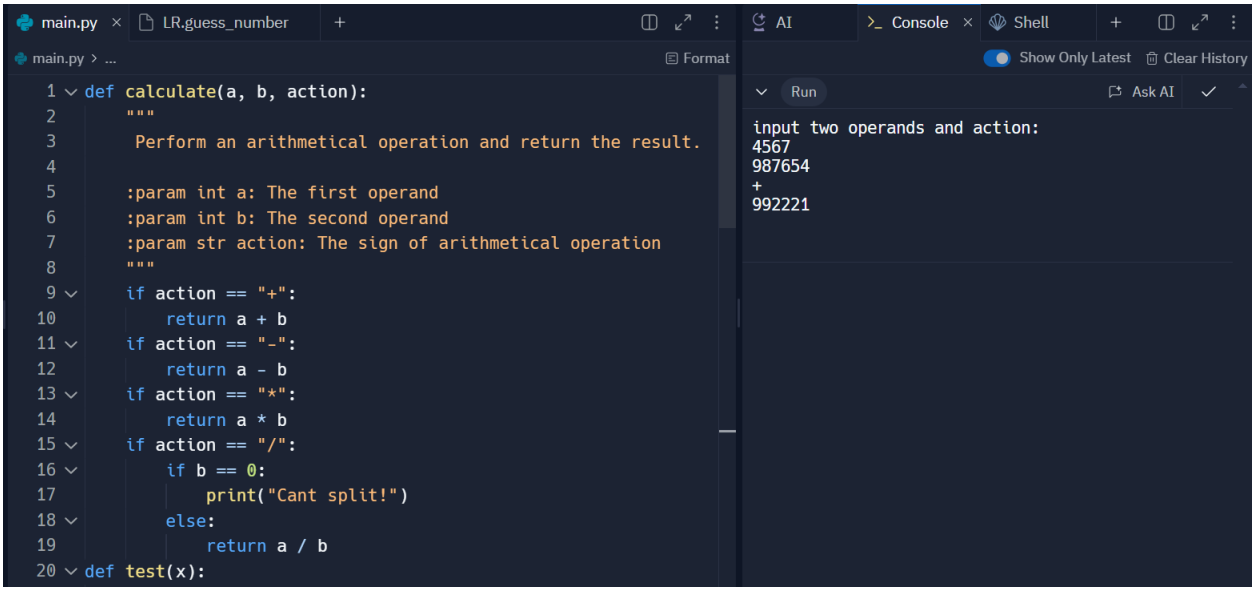
Лабораторная работа № 1.

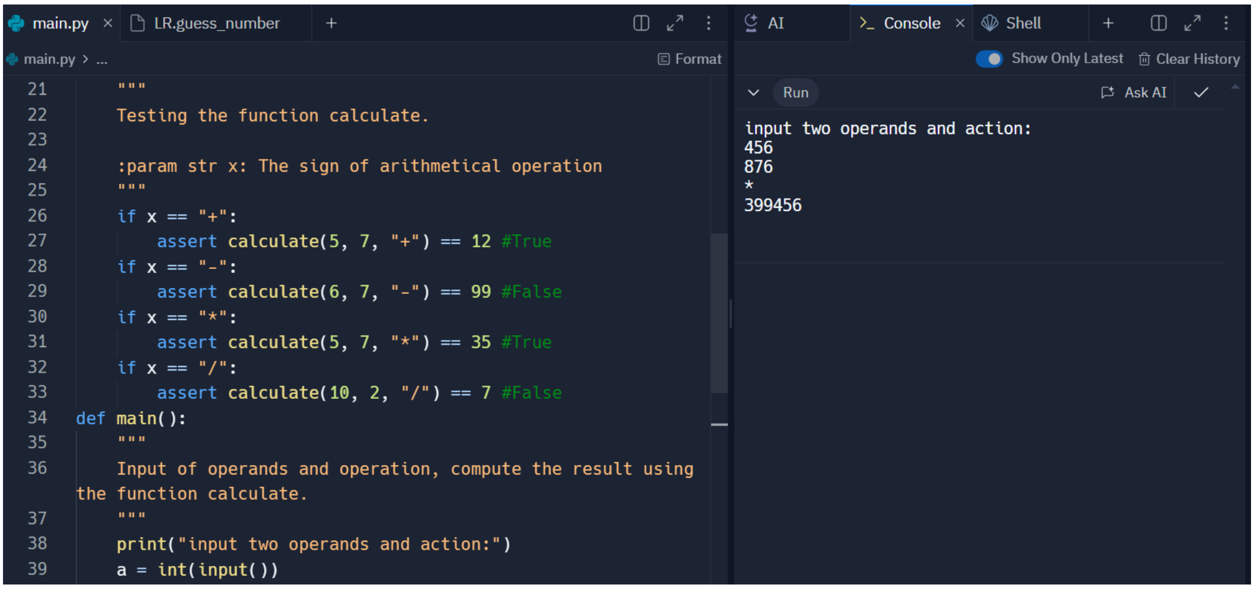
Ссылка на replit: https://replit.com/@Olljux/Herzenlabs#main.py

Ссылка на Git репозиторий: https://github.com/Olljux/Herzen_proga_2semester

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:

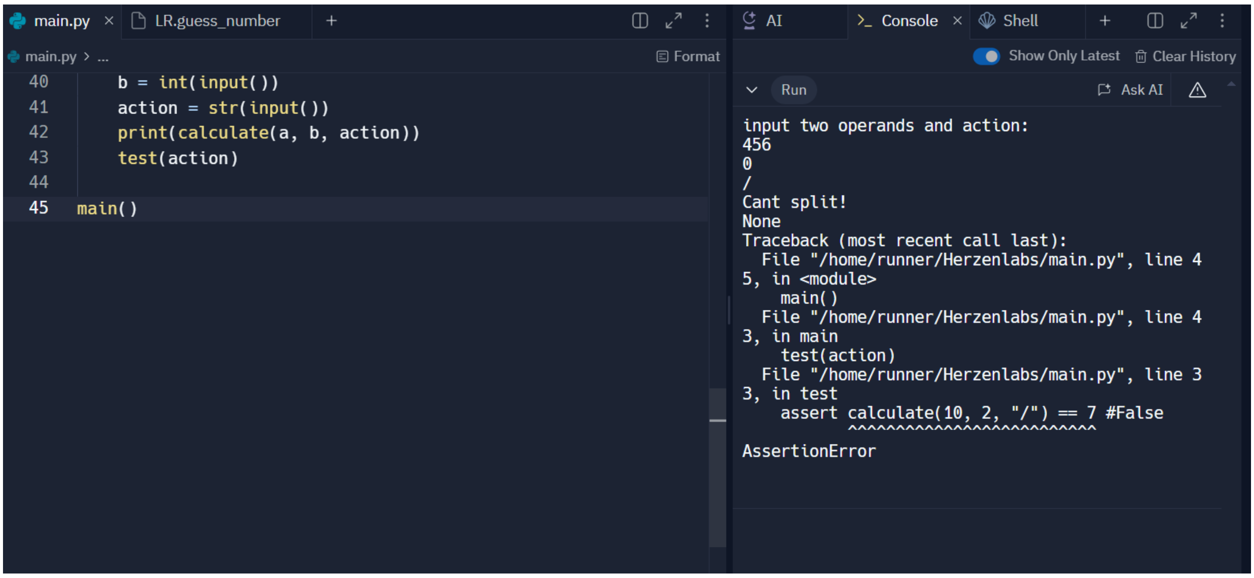




```python
    """
    Testing the function calculate.

    :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())
```

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

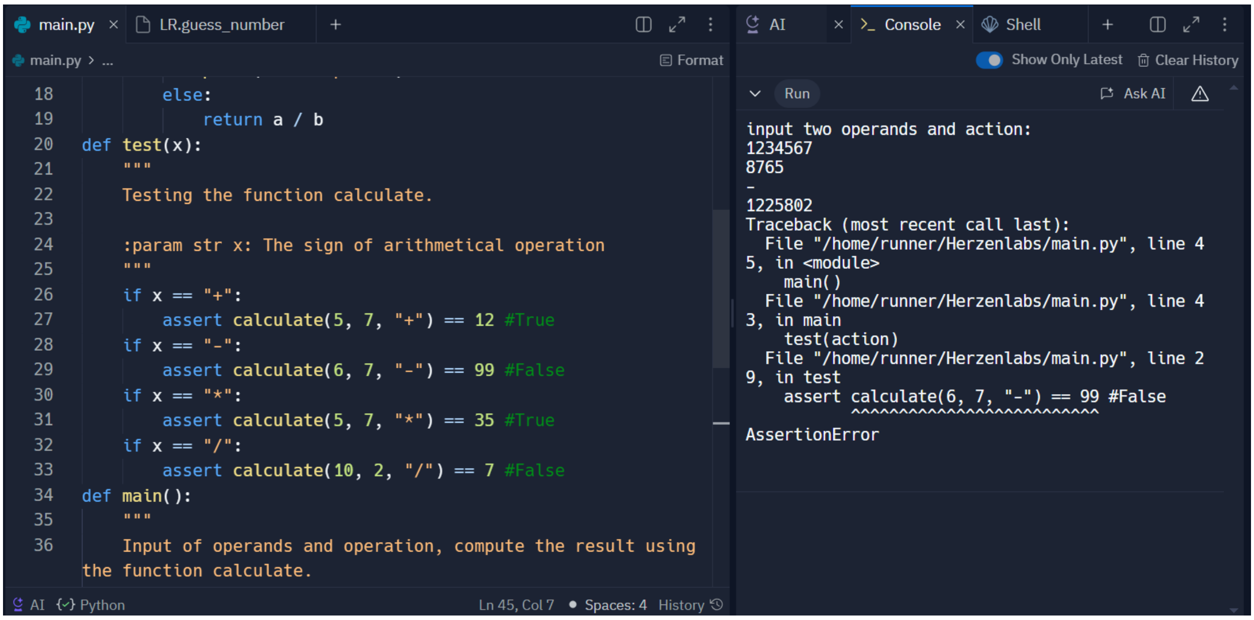


```python
    b = int(input())
    action = str(input())
    print(calculate(a, b, action))
    test(action)

main()
```

```
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
```

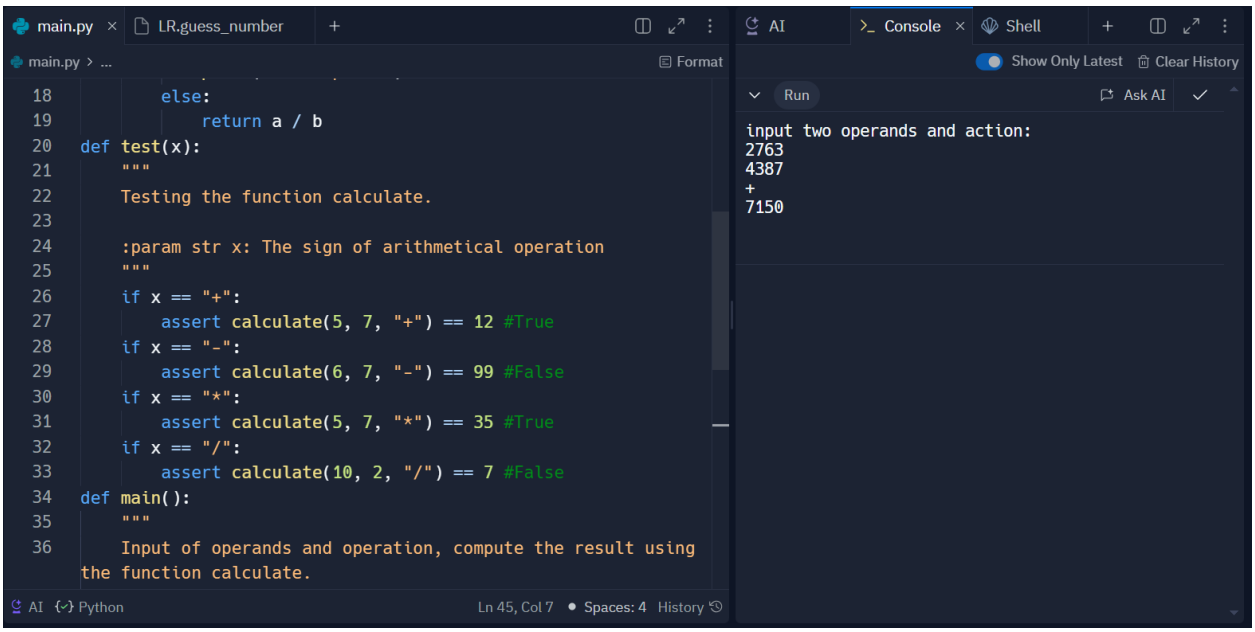
Тест с помощью assert:



```python
        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.
```

```
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
```



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 числа, 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:

```python
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:
```

```
~/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:

```python
            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()
```

```
~/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()
```