

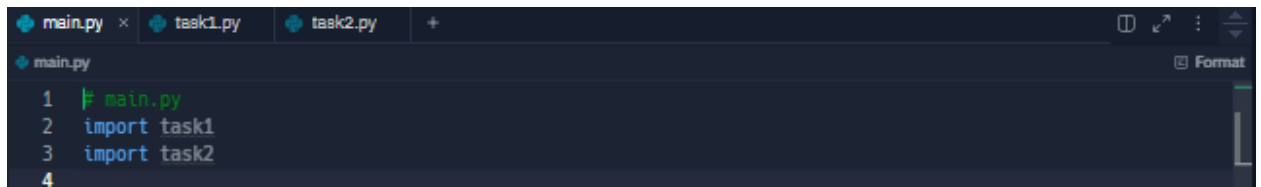
Лабораторная работа №1

Задачи.

Ссылка на Replite: <https://replit.com/@letablohina2017/Practice>

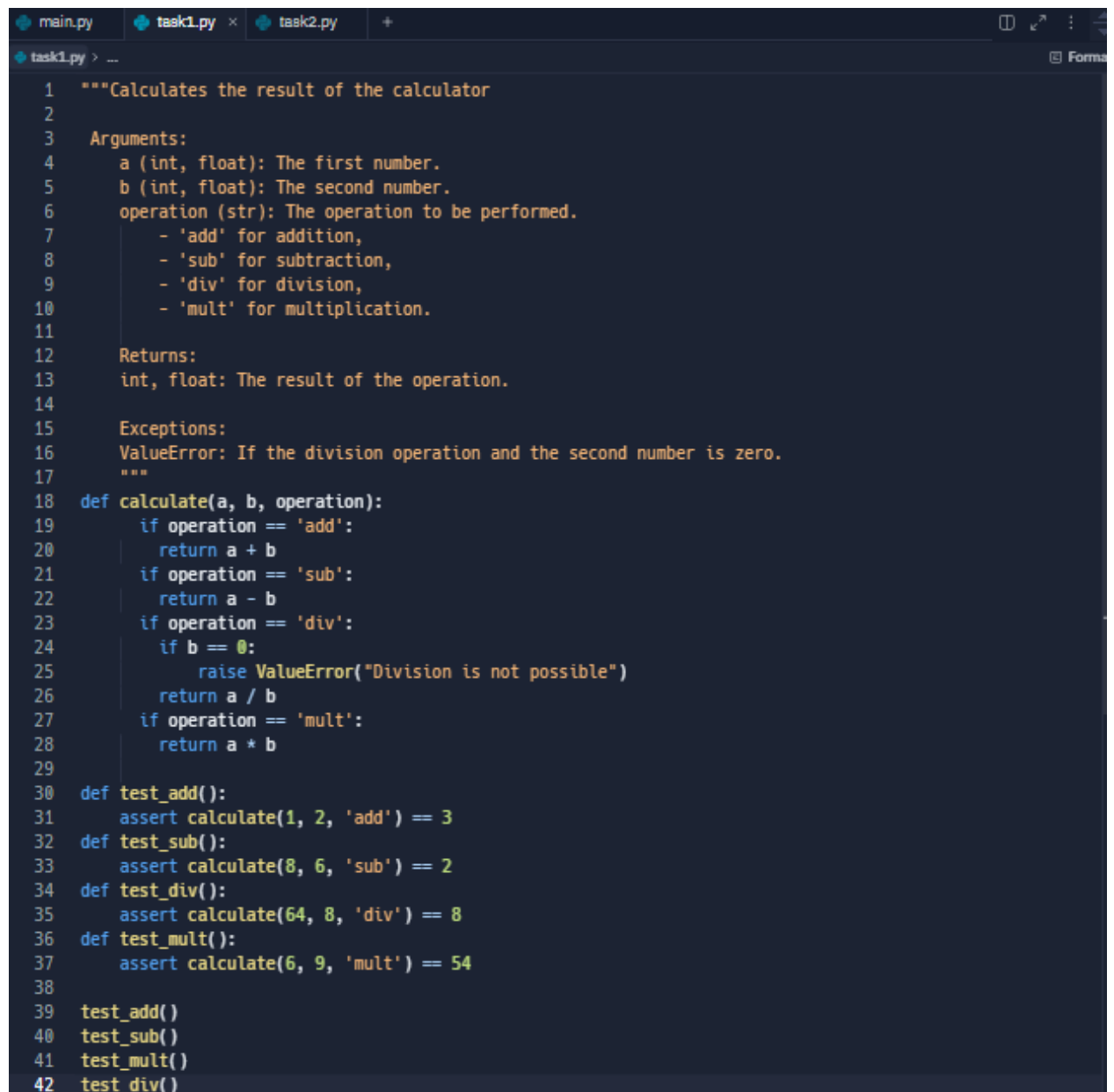
Выполнила Блохина Валерия студентка 2 курса ИВТ

Для одновременного выполнения программ в Replite были сделаны 2 файла и далее они импортированы в главный файл. Поэтому при отсутствии ошибок в работе калькулятора (1 задание), программа выводит сразу требования ввести число для 2 программы.



```
main.py x task1.py task2.py +
main.py
1 | # main.py
2 | import task1
3 | import task2
4 |
```

Задание 3.1



```
main.py task1.py x task2.py +
task1.py > ...
1 | """Calculates the result of the calculator
2 |
3 | Arguments:
4 |     a (int, float): The first number.
5 |     b (int, float): The second number.
6 |     operation (str): The operation to be performed.
7 |         - 'add' for addition,
8 |         - 'sub' for subtraction,
9 |         - 'div' for division,
10 |         - 'mult' for multiplication.
11 |
12 | Returns:
13 |     int, float: The result of the operation.
14 |
15 | Exceptions:
16 |     ValueError: If the division operation and the second number is zero.
17 | """
18 | def calculate(a, b, operation):
19 |     if operation == 'add':
20 |         return a + b
21 |     if operation == 'sub':
22 |         return a - b
23 |     if operation == 'div':
24 |         if b == 0:
25 |             raise ValueError("Division is not possible")
26 |         return a / b
27 |     if operation == 'mult':
28 |         return a * b
29 |
30 | def test_add():
31 |     assert calculate(1, 2, 'add') == 3
32 | def test_sub():
33 |     assert calculate(8, 6, 'sub') == 2
34 | def test_div():
35 |     assert calculate(64, 8, 'div') == 8
36 | def test_mult():
37 |     assert calculate(6, 9, 'mult') == 54
38 |
39 | test_add()
40 | test_sub()
41 | test_mult()
42 | test_div()
```

Задание 3.2

```
main.py x task1.py x task2.py x +
task2.py + ...
1 """The function takes a number and a range (low and high values) and tries to guess a number using
  binary search.
2 Returns the guessed number and the number of attempts, which were required for the search. If the
  number is not found in the range, returns None and the number of attempts.
3 Guessing arguments:
4 number (int): The number that needs to be guessed.
5 low (int): The lower limit of the search range.
6 high (int): The upper limit of the search range.
7 The program displays the desired number and the number of attempts
8 """
9 def guessing(number, low, high):
10     attempts = 0
11     while low <= high:
12         attempts += 1
13         middle = (low + high) // 2
14         if middle < number:
15             low = middle + 1
16         elif middle > number:
17             high = middle
18         else:
19             return middle, attempts
20     return None, attempts
21
22
23 target_number = int(input("Guess the number from 1 to 100: "))
24
25 while target_number < 1 or target_number > 100:
26     print("Guess the number from 1 to 100.")
27     target_number = int(input("Guess the number from 1 to 100: "))
28
29 guessed_number, num_attempts = guessing(target_number, 1, 100)
30
31
32 if guessed_number is not None:
33     print(f"I guessed your number: {guessed_number} in {num_attempts} attempts!")
34 else:
35     print("Couldn't guess the number.")
```

Console

Run

Guess the number from 1 to 100: 45
I guessed your number: 45 in 7 attempts!