

Отладка

Ошибки, ломающие программу

Не стоит пытаться не ошибиться — всё равно не выйдет. Гораздо важнее научиться принимать и устранять свои промахи.

Предположим, у нас имеется следующая программа:

```
1  a=3
2  b='14'
3  print(a+b)
4  |
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

[Running] python -u "c:\Users\max_k\Desktop\Я препод\debugging\1.py"

Traceback (most recent call last):

File "c:\Users\max_k\Desktop\Я препод\debugging\1.py", line 3, in <module>

print(a+b)

TypeError: unsupported operand type(s) for +: 'int' and 'str'

Мы её запустили и получили ошибку. Информация об это отображается снизу на вкладке «Output».

Что именно мы там видим?

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

[Running] python -u "c:\Users\max_k\Desktop\Я препод\debugging\1.py"

Traceback (most recent call last):

File "c:\Users\max_k\Desktop\Я препод\debugging\1.py", line 3, in <module>

Код, который не удалось выполнить

print(a+b)

Указатель на конкретное место возникновения ошибки

Путь до файла, в котором была обнаружена ошибка

Номер строки

TypeError: unsupported operand type(s) for +: 'int' and 'str'

Тип ошибки

Сообщение ошибки

Существует множество разных типов ошибок, вот самые часто встречающиеся:

TypeError — операция применена к объекту несоответствующего типа.

NameError — когда глобальная или локальная переменная не найдена.

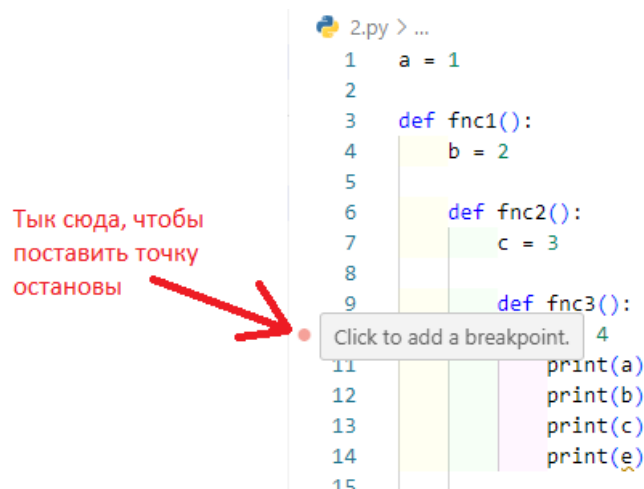
SyntaxError — когда компьютер не понимает ваш код, потому что он написан с ошибками.

Логические ошибки

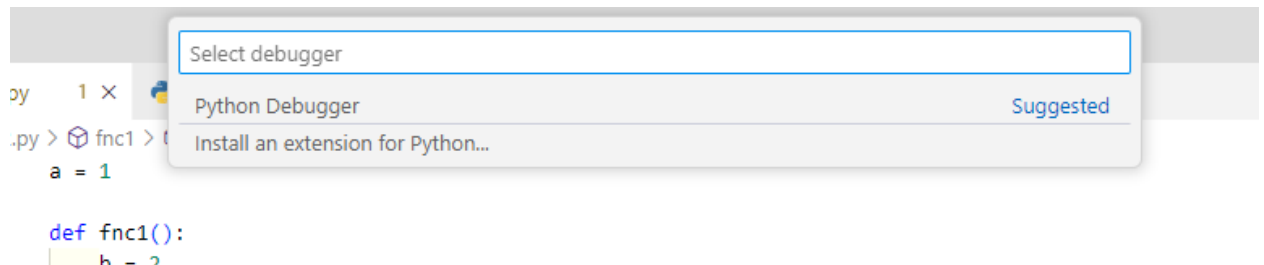
Выше мы разобрали ошибки, которые ломают код. Если их совершить, программа не будет работать. Но есть более коварный тип ошибок — логические. Они не приводят к поломке кода — он выполняется как ни в чём не бывало. Но программа выдаёт неверные результаты.

В таком случае пользуются инструментами отладки. Они позволяют отслеживать состояние программы на разных этапах выполнения. Дойдя до определённого этапа, мы можем поставить программу на паузу, чтобы проверить, в корректном ли состоянии находится программа на текущем шаге. Если всё хорошо, можем переходить к следующему шагу.

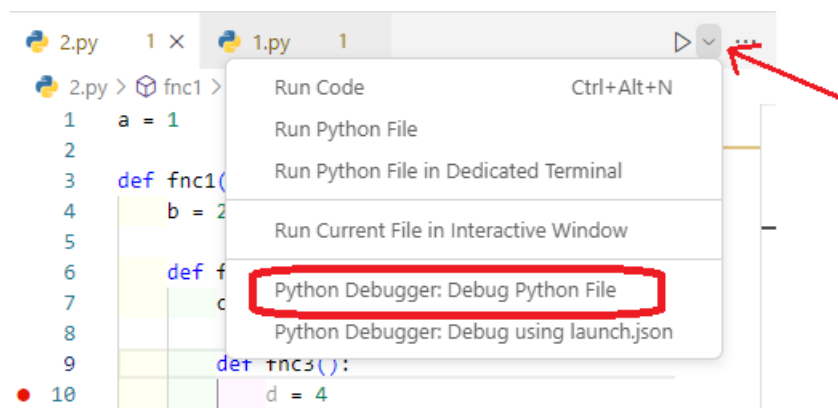
Чтобы запустить такое пошаговое выполнение программы, необходимо поставить «**точку останова**». Для этого нужно кликнуть левее номера строки, дойдя до которой, вы хотите, чтобы программа приостановила своё выполнение.



Далее нужно запустить код в «**режиме отладки**». Для этого можно нажать F5 и выбрать «Python Debugger»



Либо нажать на стрелку вниз, правее кнопки запуска, и выбрать «Python Debugger: Debug Python File».



При таком запуске программа встанет на паузу, дойдя до точки останова.

Переменные локальной области видимости функции fnc3 на текущем шаге

Переменные глобальной области видимости на текущем шаге

Номер строки, на которой приостановлена программа, в каждой из функций

Функции, которые сейчас вызваны

Кнопка, останавливающая выполнение программы

Кнопка перехода на следующий шаг

Кнопка перехода к следующей точке останова

```
1 a = 1
2
3 def fnc1():
4     b = 2
5
6     def fnc2():
7         c = 3
8
9         def fnc3():
10            d = 4
11            print(a)
12            print(b)
13            print(c)
14            print(e)
15
16        fnc3()
17
18    fnc2()
19
20 fnc1()
21
```

Когда программа приостановлена, можно ещё использовать «DEBUG CONSOLE» для работы с текущими переменными локальной или глобальной области видимости. Или даже добавить новую переменную в локальную область видимости.

← e=5

```
1 a = 1
2
3 def fnc1():
4     b = 2
5
6     def fnc2():
7         c = 3
8
9         def fnc3():
10            d = 4
11            print(a)
12            print(b)
13            print(c)
14            print(e)
15            pass
16
17        fnc3()
18
19    fnc2()
20
21 fnc1()
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE

→ print(a+b+c)

6