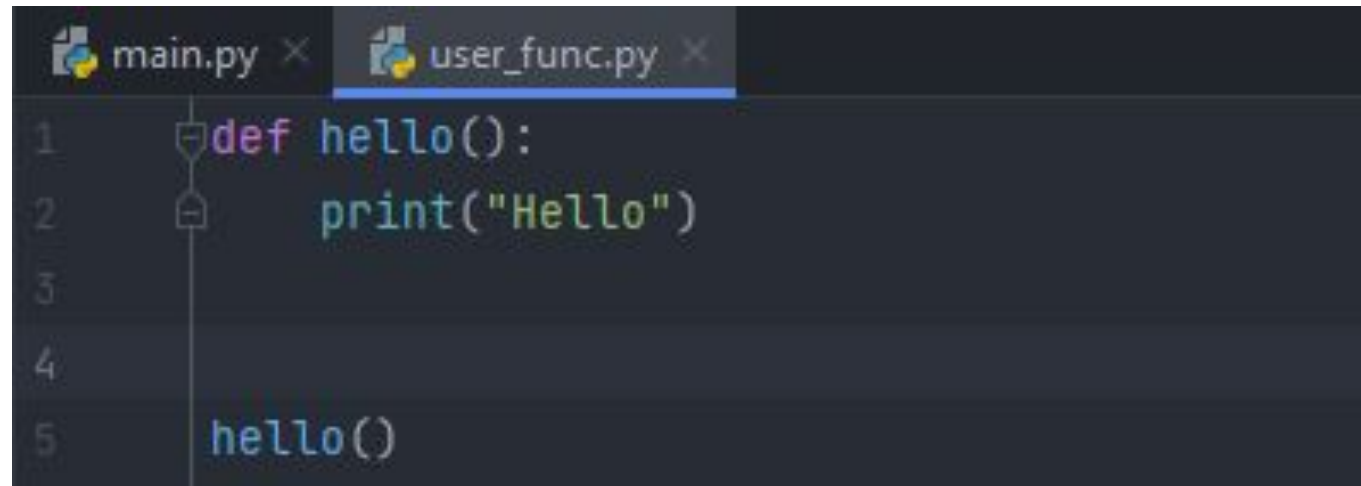


Python

Изучение функций

Пользовательские функции

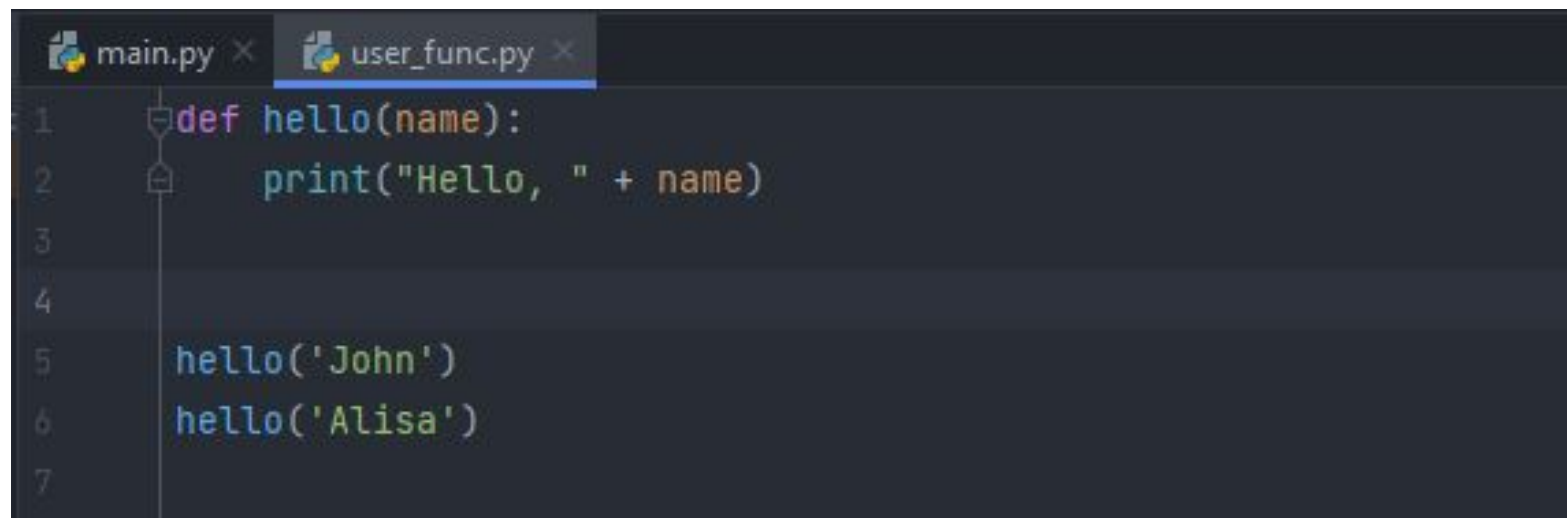
- Создание функций. Для чего это вообще может потребоваться?
- Функции в основном нужны для того, чтобы избежать дублирования кода. Чтобы многократно использовать тот или иной блок кода.



```
main.py x user_func.py x
1 def hello():
2     print("Hello")
3
4
5 hello()
```

Пользовательские функции

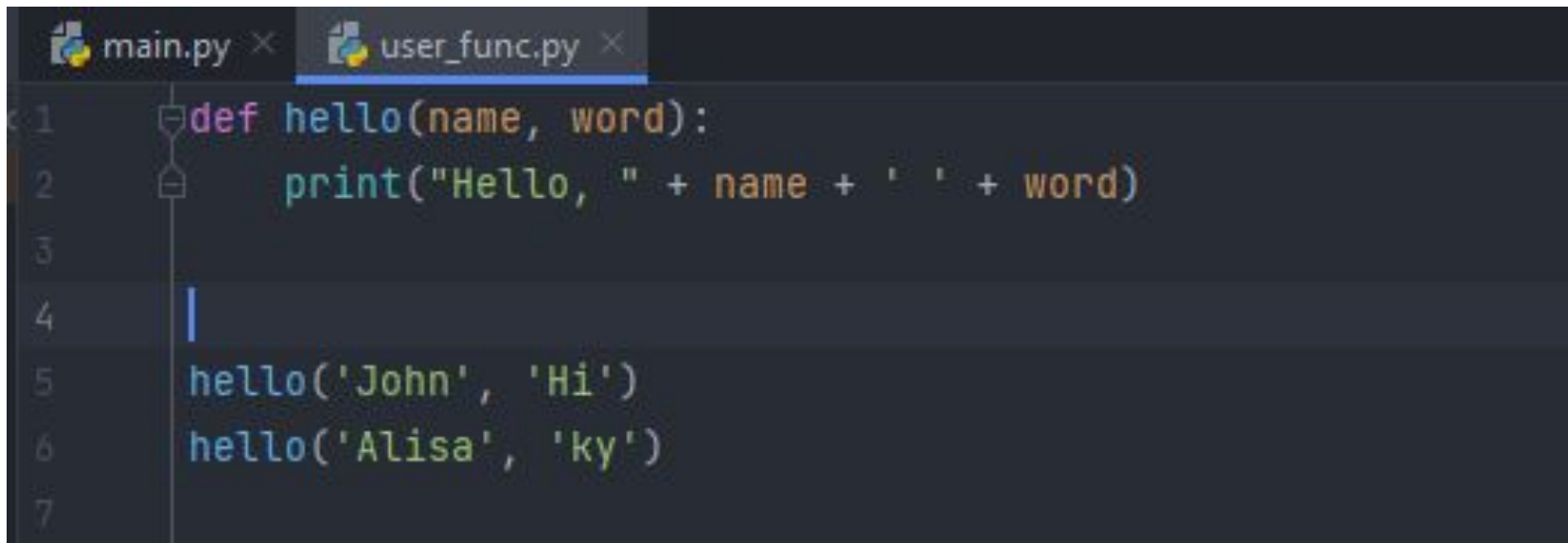
- Теперь сделаем нашу функцию более полезной. Для этого передадим ей параметр.
- Конечно же, называем наш параметр как-либо осмысленно.



```
main.py x user_func.py x
1 def hello(name):
2     print("Hello, " + name)
3
4
5 hello('John')
6 hello('Alisa')
7
```

Пользовательские функции

- Соответственно, может добавить второй параметр. Например, «word». Выглядеть это будет следующим образом, и, разумеется, его надо будет указать на вызове функции.



```
main.py × user_func.py ×  
1 def hello(name, word):  
2     print("Hello, " + name + ' ' + word)  
3  
4 |  
5 hello('John', 'Hi')  
6 hello('Alisa', 'ky')  
7
```

Пользовательские функции

- Например, при написании функции «def sum» - интерпретатор питона подчеркнёт нам слово sum.
- Всё дело в том, что функции в питоне работают несколько иначе, чем функции, например, в том же php.
- В php, если мы создадим функцию или если в php встроена какая-то функция, и мы попытаемся создать аналогичную функцию с таким же именем, то php нам это сделать банально не позволит.
- Он скажет, что такая функция уже задекларирована, определена и мы не можем переопределять функции.

Пользовательские функции

- В питоне... А что в питоне...
- В питоне это возможно. Но делать так не советуют. Более того, с функциями, которые уже встроены в питоне. Как раз такая, как функция `sum`.

A screenshot of a Python IDE with a dark theme. The top bar shows two tabs: 'main.py' and 'user_func.py'. The 'user_func.py' tab is active and shows the following code:

```
1 def sum():  
2     print("Test")  
3  
4  
5 sum()  
6
```

The function name 'sum' is underlined with a wavy line, indicating a shadowed name. A lightbulb icon is visible next to the function call 'sum()' on line 5, suggesting an IDE warning or suggestion. The cursor is positioned at the end of the function call.

Пользовательские функции

- Разумеется, если я всё таки такую функцию заюзаю, то она создастся и будет использоваться. Но опять таки такая тема не рекомендуется к использованию вообще ни разу.
- Например потому что встроенную функцию `sum` уже использовать нельзя будет, так как мы её банально переопределили 😊
- Поэтому в подобных случаях называть функции стоит как-нибудь иначе, например: «`get_sum`», как вариант.

Пользовательские функции

- Пусть наша сумма будет принимать два аргумента: «a» и «b».
- Опять таки, называть можно как угодно и как вы захотите. Главное, чтобы в теле функции вы должны к ним обращаться точно так же, как определили в виде аргументов в начале.
- При вызове функции мы, соответственно, можем передать аргументы. (1, 3), например.
- Ну либо же можно просто задать x и y со значениями, затем обратиться к функции, указав x и y в качестве принимаемых аргументов.

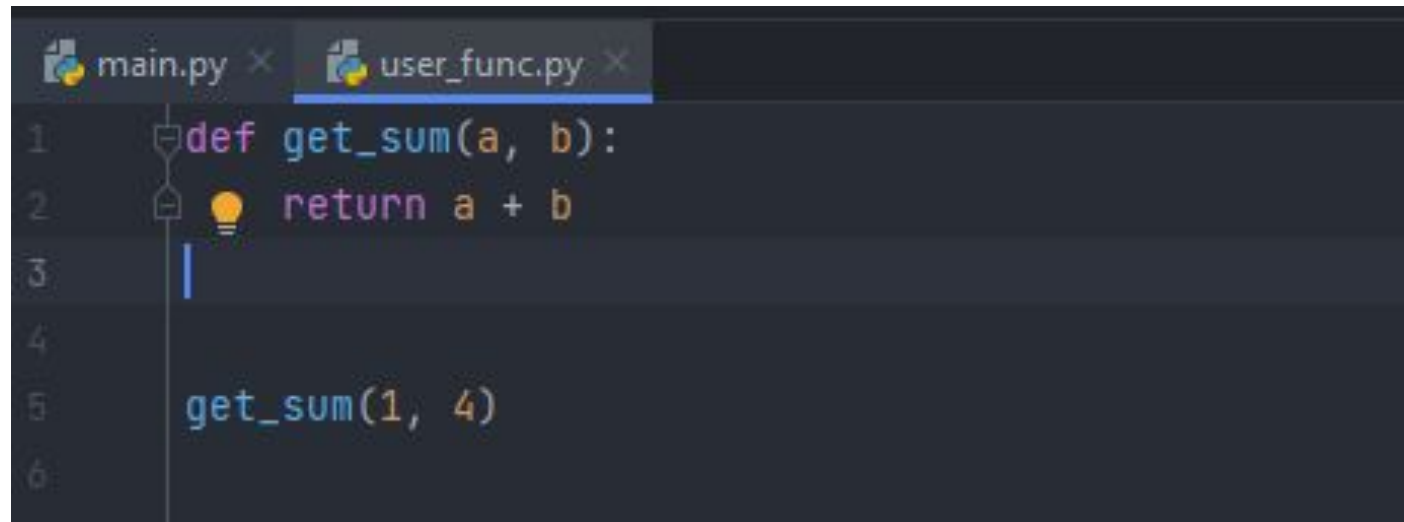
Пользовательские функции

- Пример:

```
main.py x user_func.py x
1 def get_sum(a, b):
2     print(a + b)
3
4
5     x = 4
6     y = 4
7
8     get_sum(1, 3)
9     get_sum(x, y)
10
```

Пользовательские функции

- А если попробовать так?



```
main.py x user_func.py x
1 def get_sum(a, b):
2     return a + b
3
4
5 get_sum(1, 4)
6
```

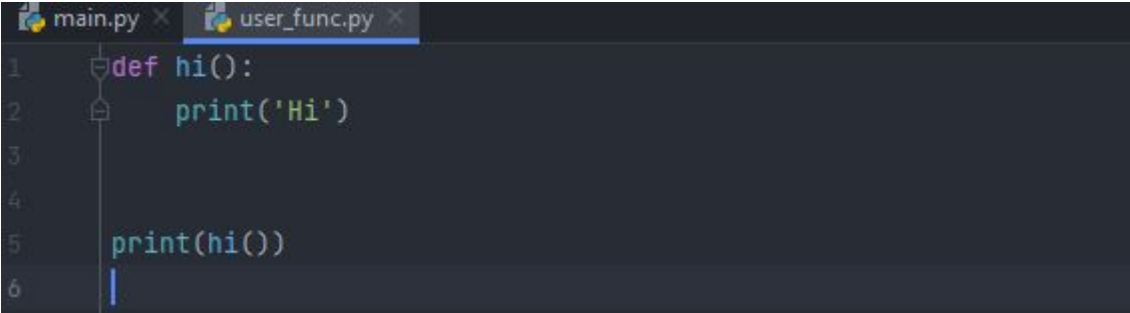
- В чём ошибка и как сделать так, чтобы выводило.

Пользовательские функции

- Почему стоит возвращать результат?
- Потому что нам не всегда нужно его печатать. Вполне вероятно, что мы можем делать какие-нибудь промежуточные вычисления и желаем использовать в дальнейших вычислениях.

Пользовательские функции

- На что еще стоит обратить внимание?

- 

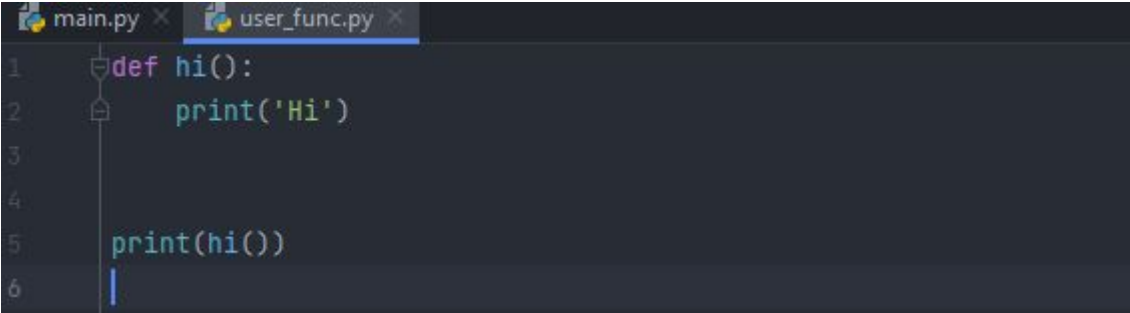
```
main.py x user_func.py x
1 def hi():
2     print('Hi')
3
4
5 print(hi())
6
```

Потому что функция всегда что-то неявно возвращает.

- Когда мы не используем `print()` – этот результат подавляется, он не выводится.
- Нюанс помнить – если ваша функция что-то печатает, то печатать дополнительно ничего не нужно.

Пользовательские функции

- На что еще стоит обратить внимание?

- 

```
main.py x user_func.py x
1 def hi():
2     print('Hi')
3
4
5 print(hi())
6
```

Потому что функция всегда что-то неявно возвращает.

- Когда мы не используем `print()` – этот результат подавляется, он не выводится.
- Нюанс помнить – если ваша функция что-то печатает, то печатать дополнительно ничего не нужно.