при поддержке

{а}вижн

Лекция 2

в которой мы поговорим про имена

Перевод и адаптация материалов Джона ДеНиро (John DeNero). Используется с разрешения автора.

Имена, присвоения и пользовательские функции

(Пример)

Виды выражений

Виды выражений

Простые выражения:

Виды выражений

Простые выражения:

2

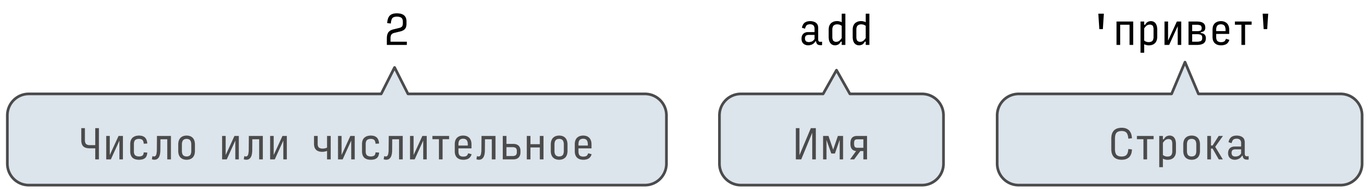
Число или числительное

Виды выражений

Простые выражения:

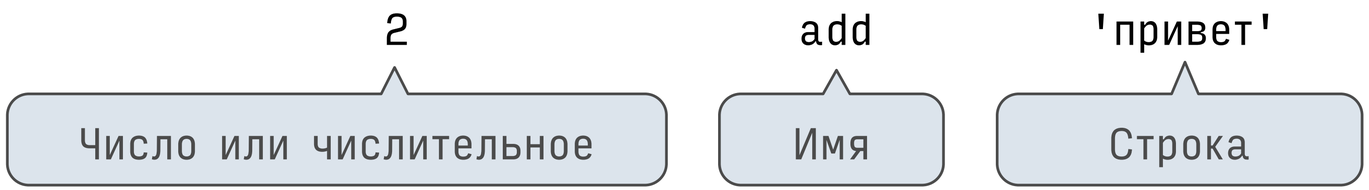
Виды выражений

Простые выражения:



Виды выражений

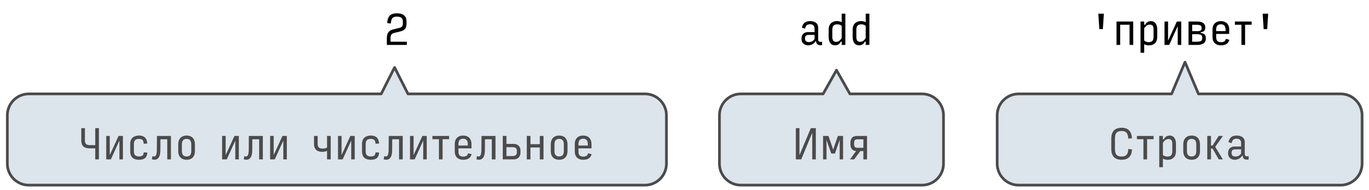
Простые выражения:



Вызывающие выражения:

Виды выражений

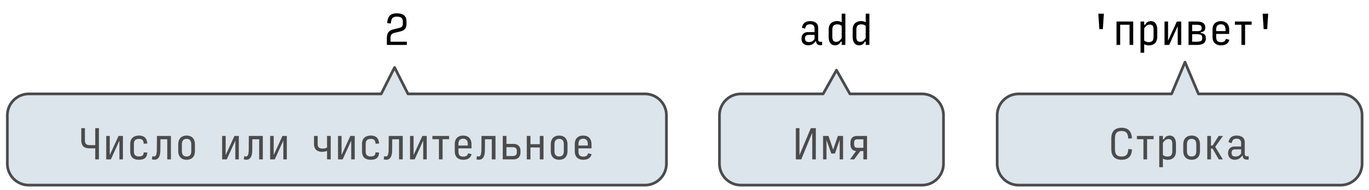
Простые выражения:



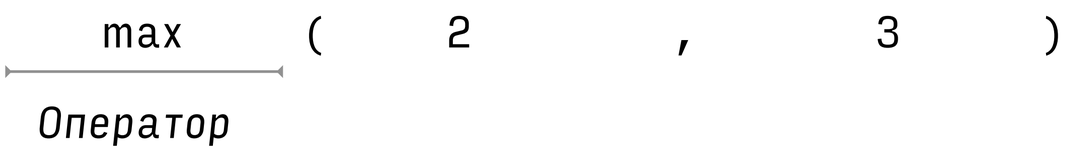
Вызывающие выражения: `max ( 2 , 3 )`

Виды выражений

Простые выражения:

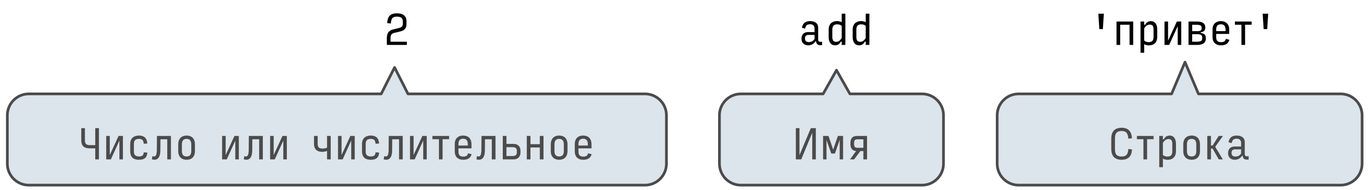


Вызывающие выражения:

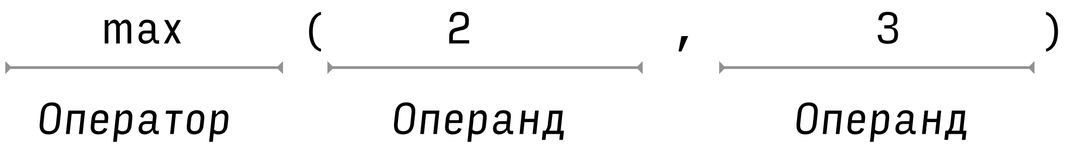


Виды выражений

Простые выражения:

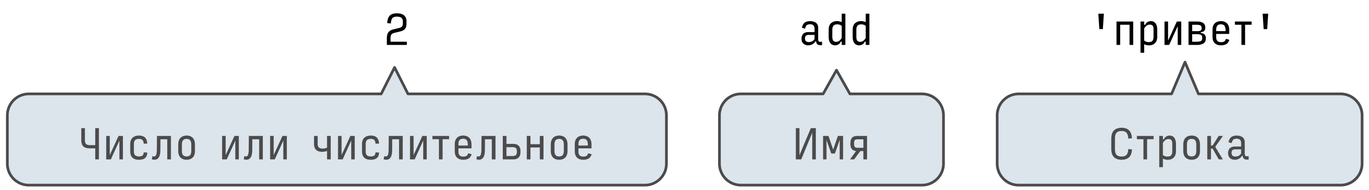


Вызывающие выражения:

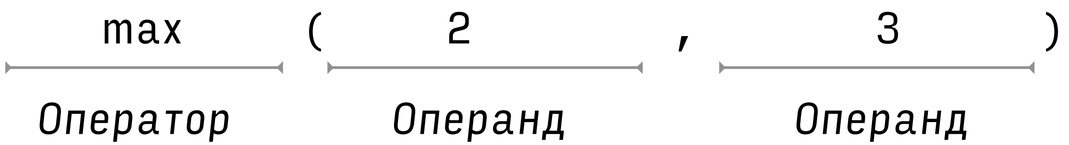


Виды выражений

Простые выражения:



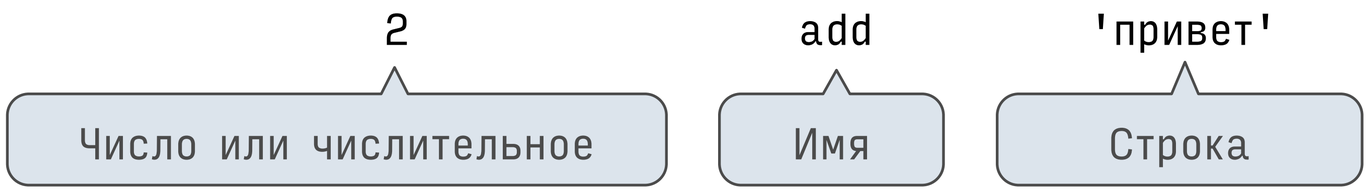
Вызывающие выражения:




```
max(min(pow(3, 5), -4), min(1, -2))
```

Виды выражений

Простые выражения:



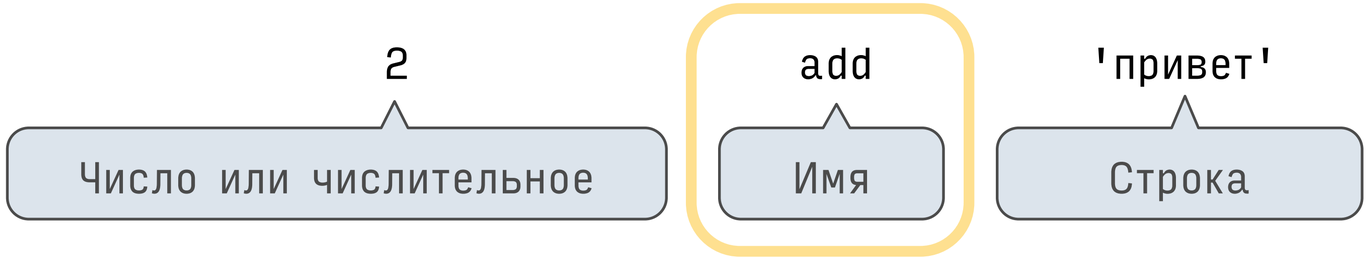
Вызывающие выражения:

Операнд также может быть вызывающим выражением

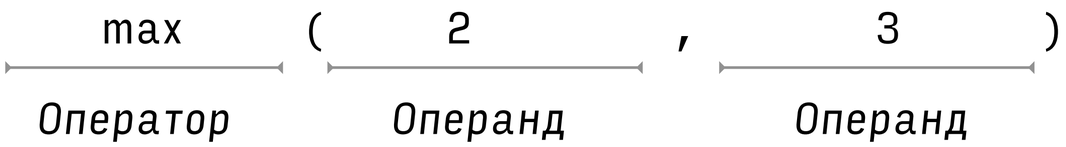
```
max(min(pow(3, 5), -4), min(1, -2))
```

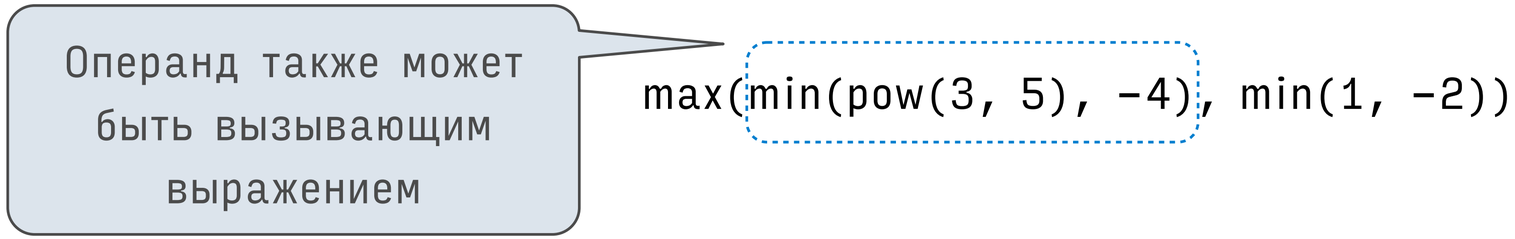
Виды выражений

Простые выражения:



Вызывающие выражения:





Вопрос 1

Вопрос 1

Каков результат последнего выражения в такой последовательности?

Вопрос 1

Каков результат последнего выражения в такой последовательности?

```
>>> f = min
```

Вопрос 1

Каков результат последнего выражения в такой последовательности?

```
>>> f = min

>>> f = max
```

Вопрос 1

Каков результат последнего выражения в такой последовательности?

```
>>> f = min

>>> f = max

>>> g, h = min, max
```

Вопрос 1

Каков результат последнего выражения в такой последовательности?

```
>>> f = min

>>> f = max

>>> g, h = min, max

>>> max = g
```

Вопрос 1

Каков результат последнего выражения в такой последовательности?

```
>>> f = min

>>> f = max

>>> g, h = min, max

>>> max = g

>>> max(f(2, g(h(1, 5), 3)), 4)
```

Вопрос 1

Каков результат последнего выражения в такой последовательности?

```
>>> f = min

>>> f = max

>>> g, h = min, max

>>> max = g

>>> max(f(2, g(h(1, 5), 3)), 4)
```

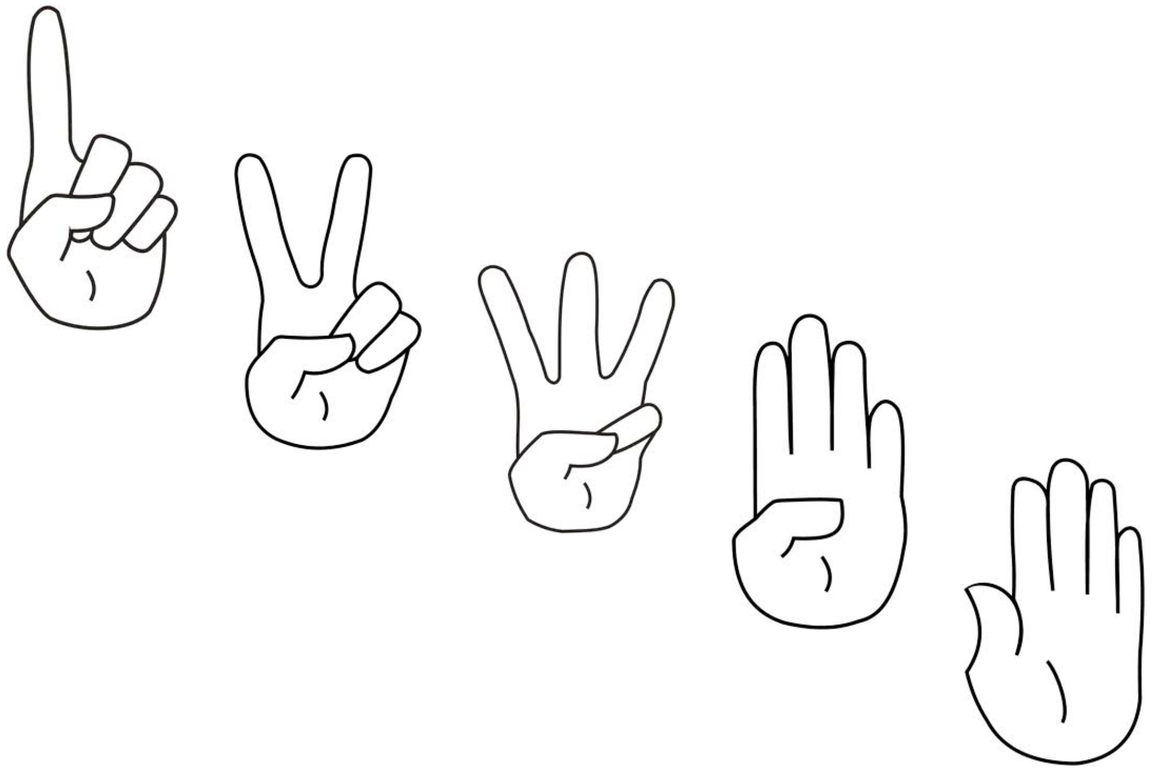
???

Вопрос 1

Каков результат последнего выражения в такой последовательности?

```
>>> f = min
>>> f = max
>>> g, h = min, max
>>> max = g
>>> max(f(2, g(h(1, 5), 3)), 4)
```

???



Диаграммы окружения

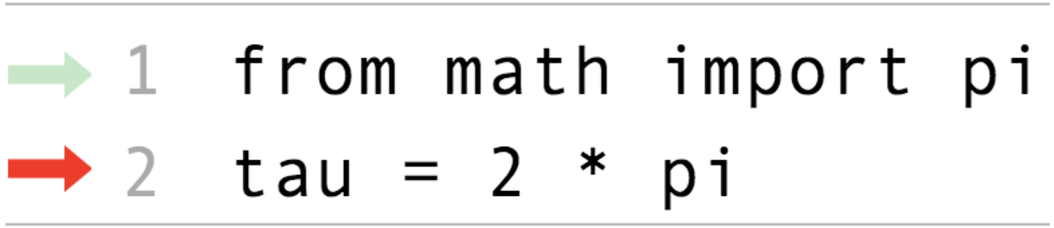
Диаграммы окружения

Диаграммы окружения визуализируют процесс интерпретации.

Интерактивная диаграмма

Диаграммы окружения

Диаграммы окружения визуализируют процесс интерпретации.

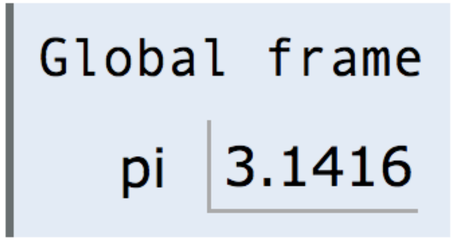


Интерактивная диаграмма

Диаграммы окружения

Диаграммы окружения визуализируют процесс интерпретации.

```
1  from math import pi
2  tau = 2 * pi
```

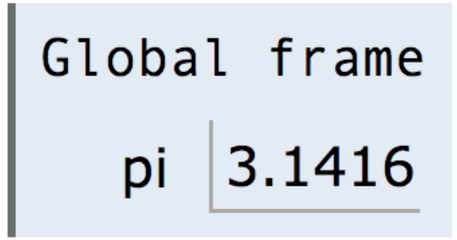


Интерактивная диаграмма

Диаграммы окружения

Диаграммы окружения визуализируют процесс интерпретации.

```
1  from math import pi
2  tau = 2 * pi
```



Код (слева):

Фреймы (справа):

Интерактивная диаграмма

Диаграммы окружения

Диаграммы окружения визуализируют процесс интерпретации.

Код (слева):

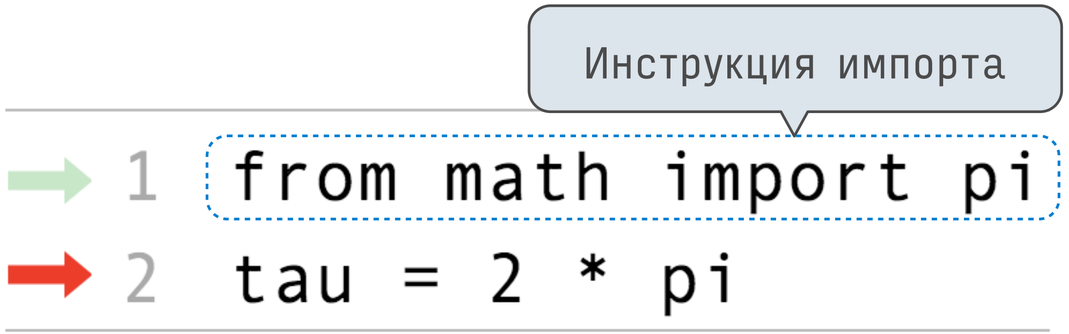
Инструкции и выражения

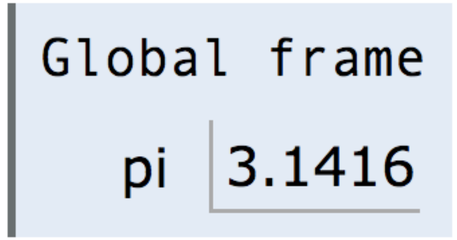
Фреймы (справа):

Интерактивная диаграмма

Диаграммы окружения

Диаграммы окружения визуализируют процесс интерпретации.





Код (слева):

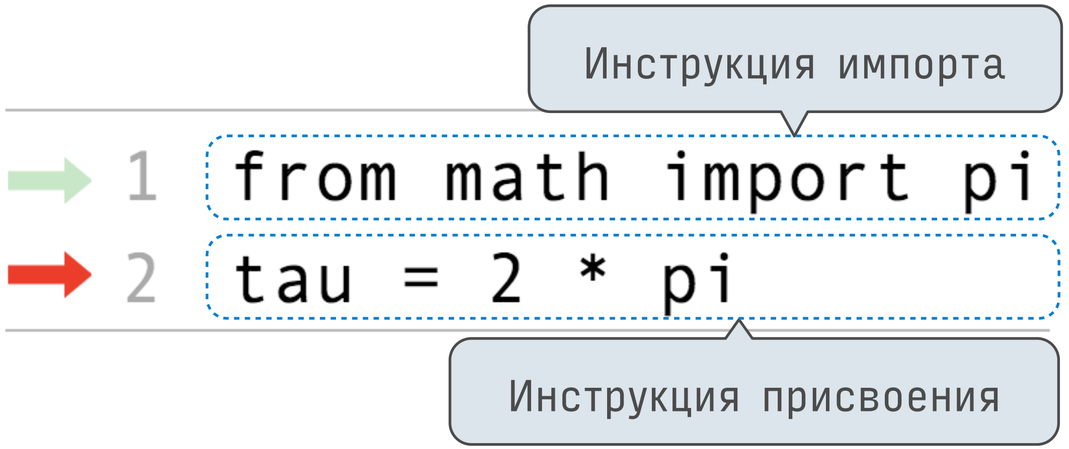
Инструкции и выражения

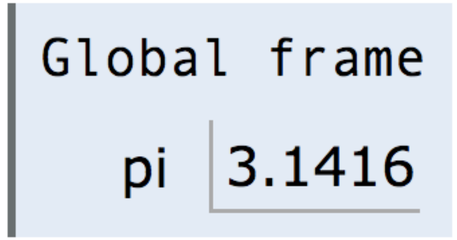
Фреймы (справа):

Интерактивная диаграмма

Диаграммы окружения

Диаграммы окружения визуализируют процесс интерпретации.





Код (слева):

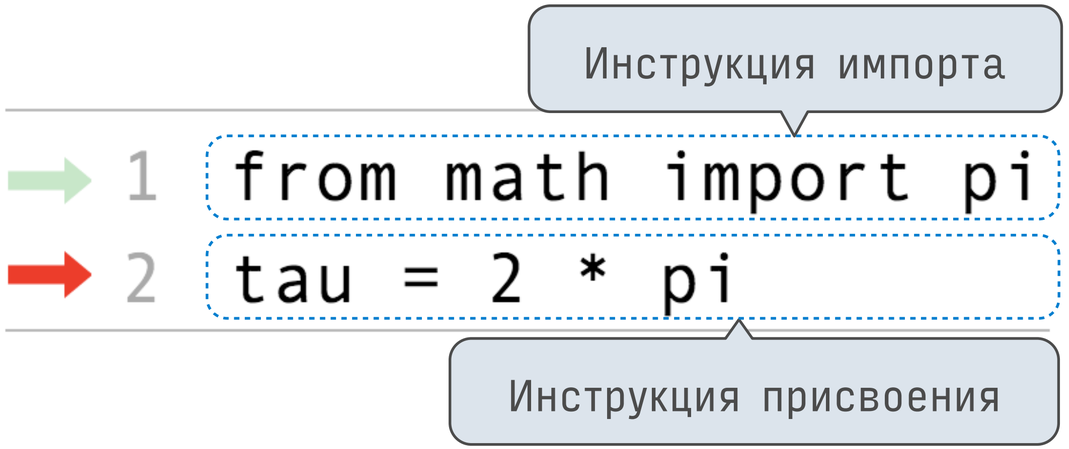
Инструкции и выражения

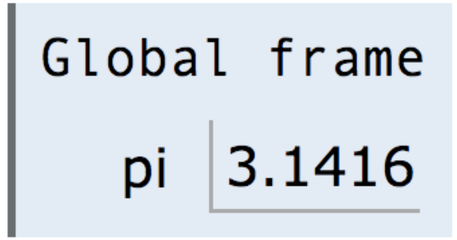
Фреймы (справа):

Интерактивная диаграмма

Диаграммы окружения

Диаграммы окружения визуализируют процесс интерпретации.





Код (слева):

Инструкции и выражения

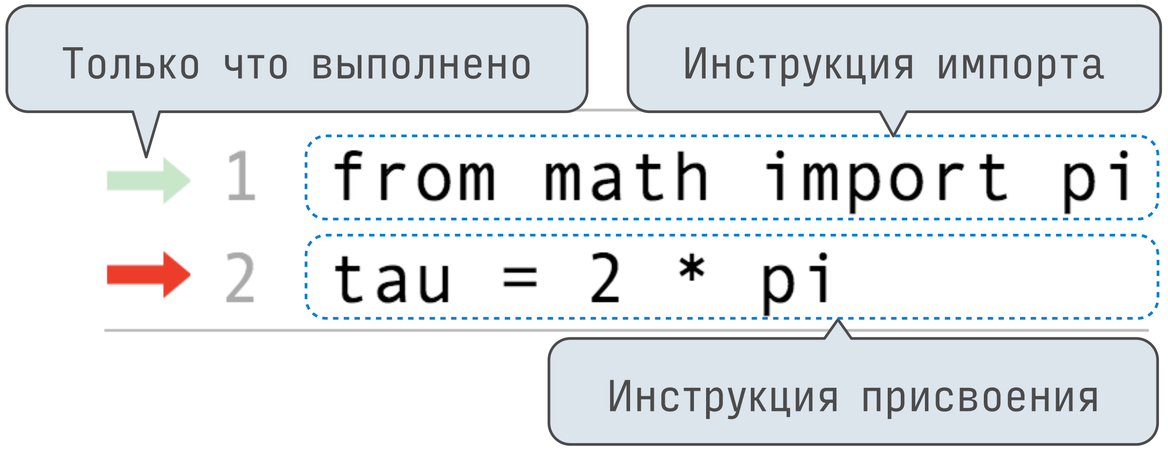
Стрелки указывают порядок исполнения

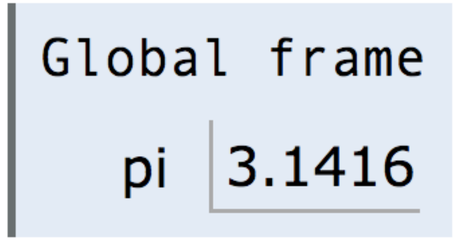
Фреймы (справа):

Интерактивная диаграмма

Диаграммы окружения

Диаграммы окружения визуализируют процесс интерпретации.





Код (слева):

Инструкции и выражения

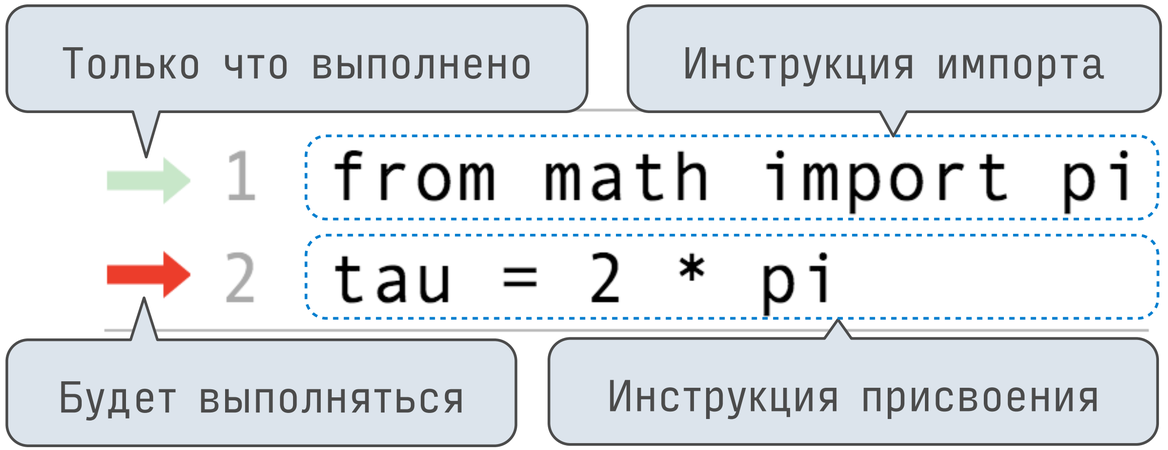
Стрелки указывают порядок исполнения

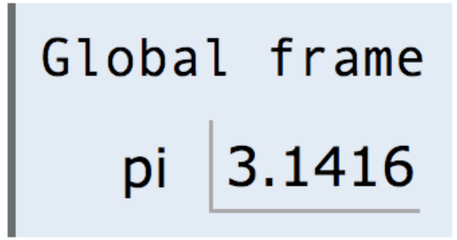
Фреймы (справа):

Интерактивная диаграмма

Диаграммы окружения

Диаграммы окружения визуализируют процесс интерпретации.





Код (слева):

Инструкции и выражения

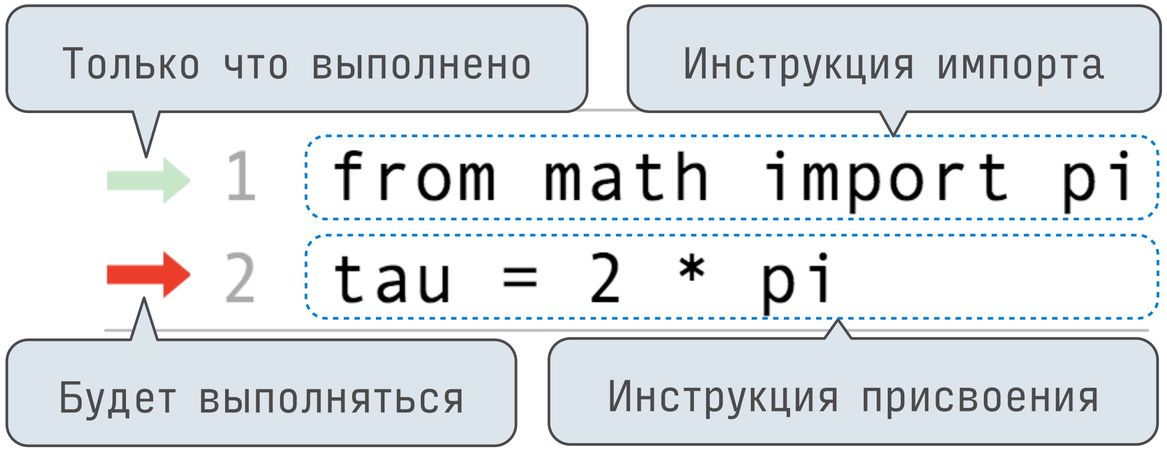
Стрелки указывают порядок исполнения

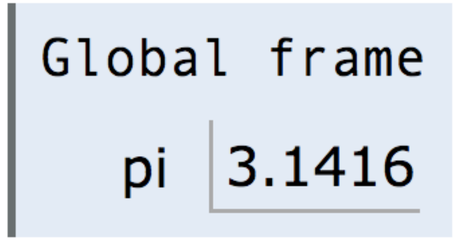
Фреймы (справа):

Интерактивная диаграмма

Диаграммы окружения

Диаграммы окружения визуализируют процесс интерпретации.





Код (слева):

Инструкции и выражения

Стрелки указывают порядок исполнения

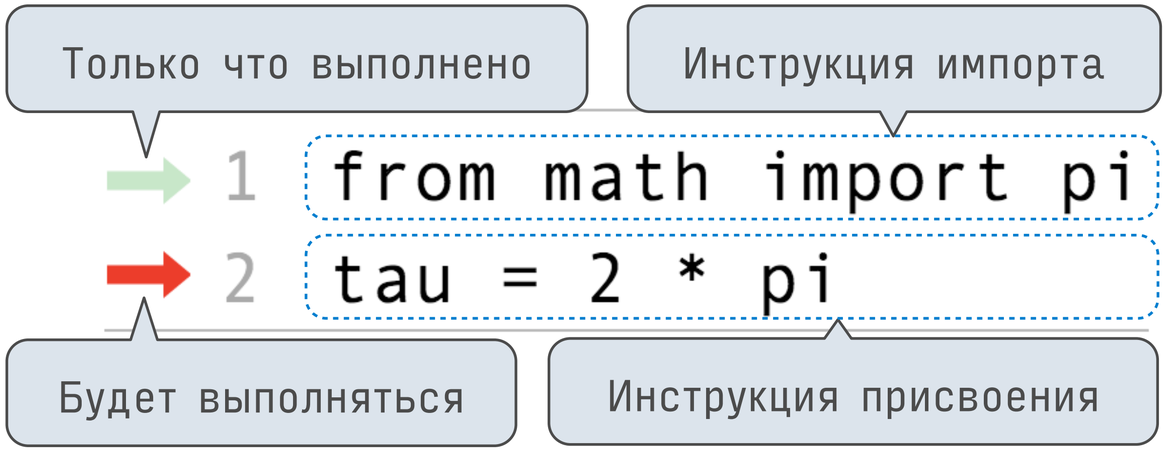
Фреймы (справа):

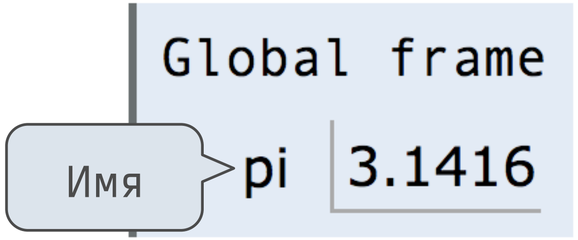
Каждое имя связано со значением

Интерактивная диаграмма

Диаграммы окружения

Диаграммы окружения визуализируют процесс интерпретации.





Код (слева):

Инструкции и выражения

Стрелки указывают порядок исполнения

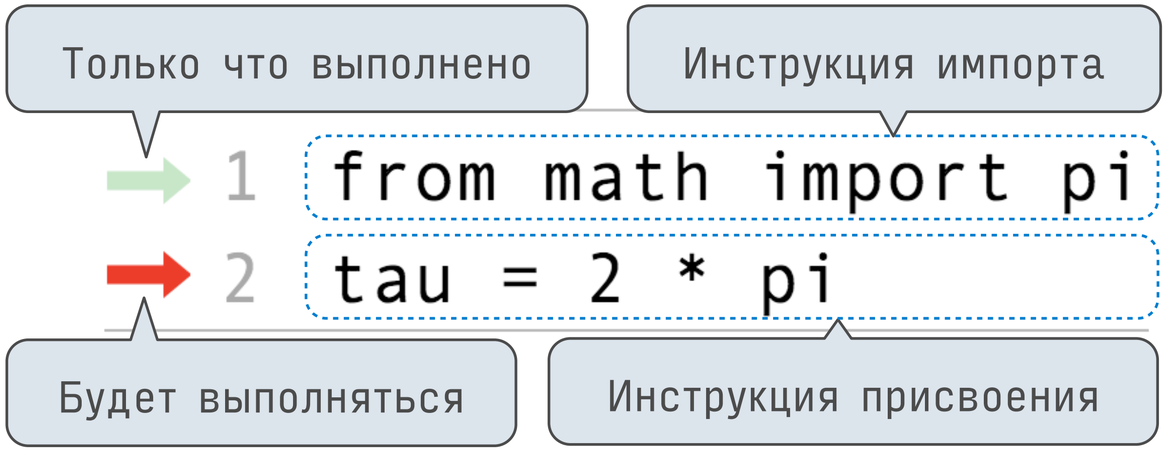
Фреймы (справа):

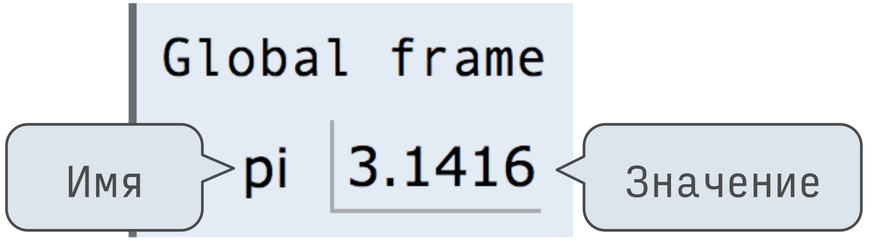
Каждое имя связано со значением

Интерактивная диаграмма

Диаграммы окружения

Диаграммы окружения визуализируют процесс интерпретации.





Код (слева):

Инструкции и выражения

Стрелки указывают порядок исполнения

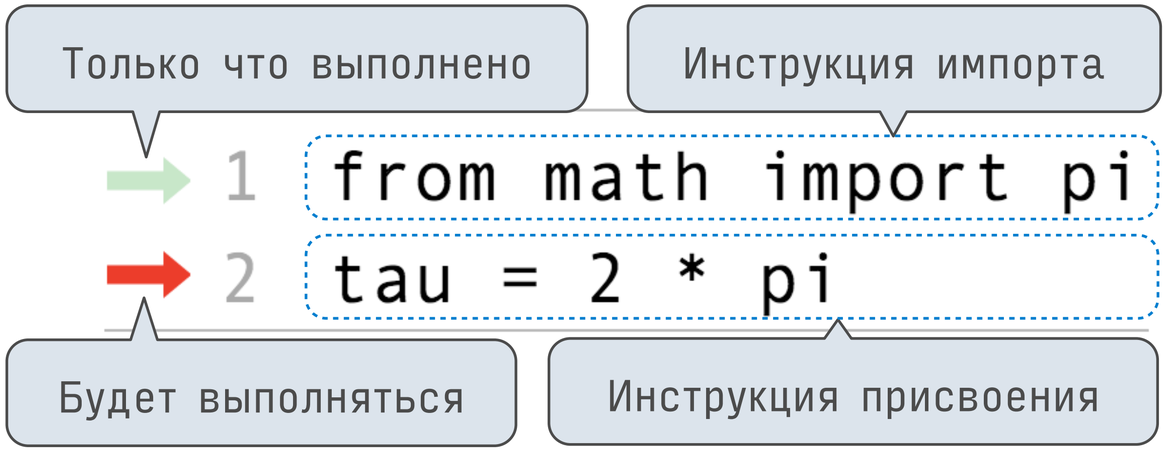
Фреймы (справа):

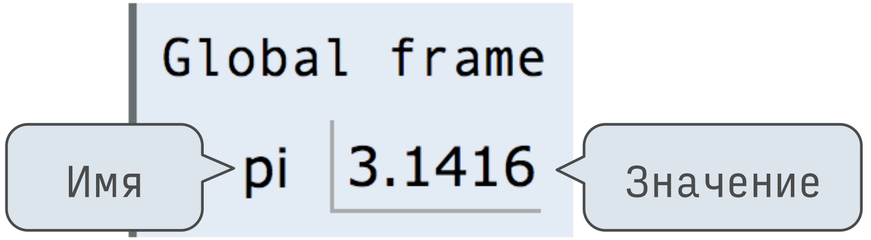
Каждое имя связано со значением

Интерактивная диаграмма

Диаграммы окружения

Диаграммы окружения визуализируют процесс интерпретации.





Код (слева):

Инструкции и выражения

Стрелки указывают порядок исполнения

Фреймы (справа):

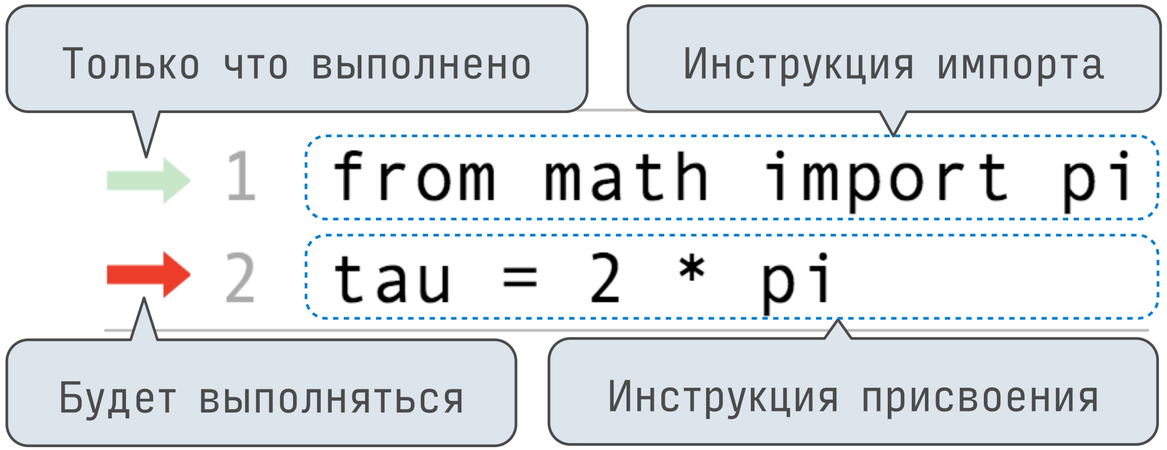
Каждое имя связано со значением

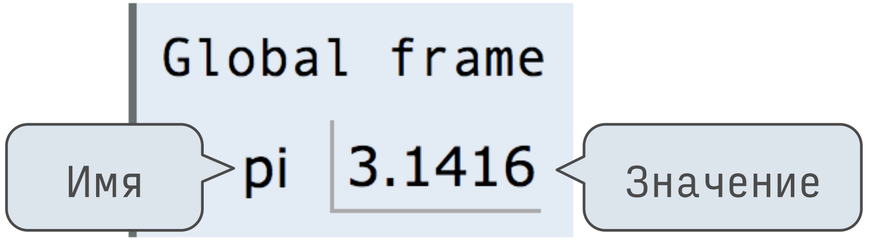
Внутри фрейма имя не может повторяться

Интерактивная диаграмма

Диаграммы окружения

Диаграммы окружения визуализируют процесс интерпретации.





Код (слева):

Инструкции и выражения

Стрелки указывают порядок исполнения

Фреймы (справа):

Каждое имя связано со значением

Внутри фрейма имя не может повторяться

(Пример)

Интерактивная диаграмма

Инструкция присвоения

Интерактивная диаграмма

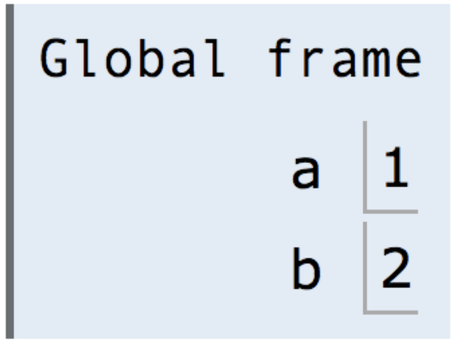
Инструкция присвоения

```
1  a = 1
2  b = 2
3  b, a = a + b, b
```

Интерактивная диаграмма

Инструкция присвоения

```
1  a = 1
2  b = 2
3  b, a = a + b, b
```

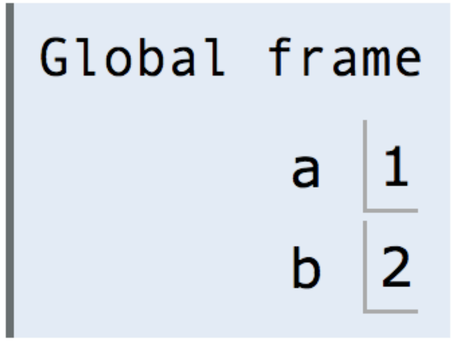


Интерактивная диаграмма

Инструкция присвоения

Только что выполнено

```
1  a = 1
2  b = 2
3  b, a = a + b, b
```



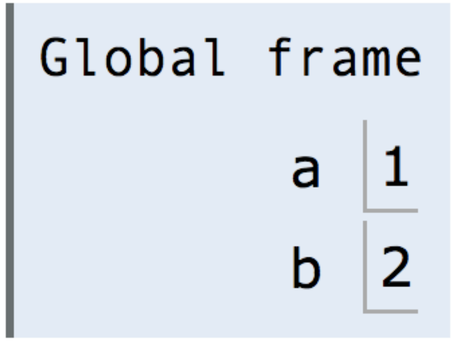
Интерактивная диаграмма

Инструкция присвоения

Только что выполнено

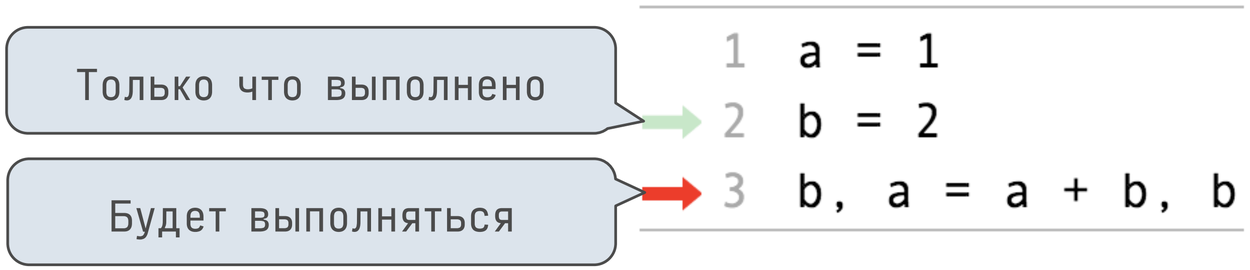
Будет выполняться

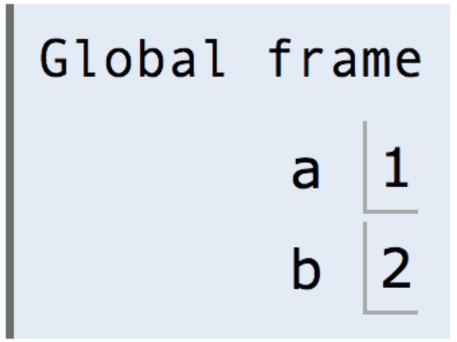
```
1  a = 1
2  b = 2
3  b, a = a + b, b
```



Интерактивная диаграмма

Инструкция присвоения

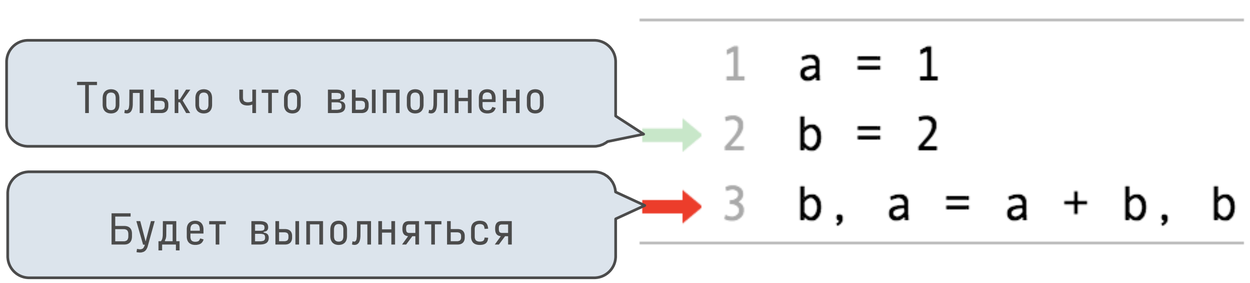


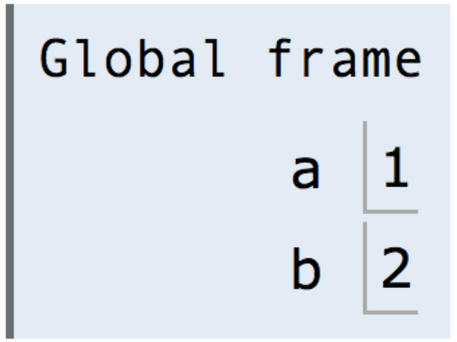


Правила выполнения инструкций присвоения:

Интерактивная диаграмма

Инструкция присвоения



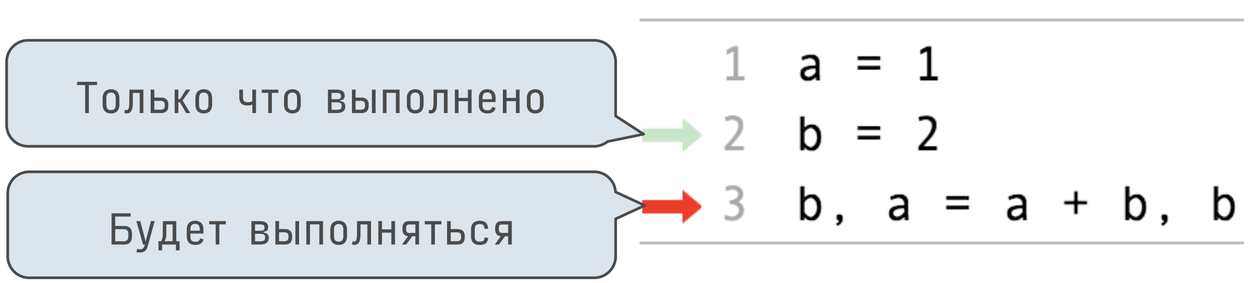


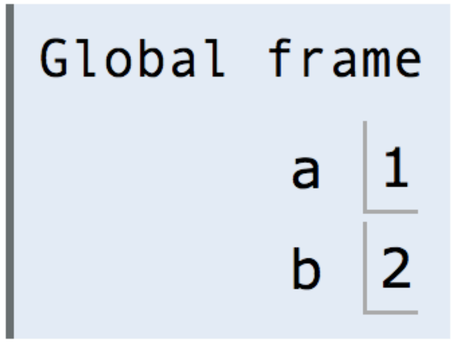
Правила выполнения инструкций присвоения:

1. Вычислить значения всех выражений справа от = слева направо.

Интерактивная диаграмма

Инструкция присвоения



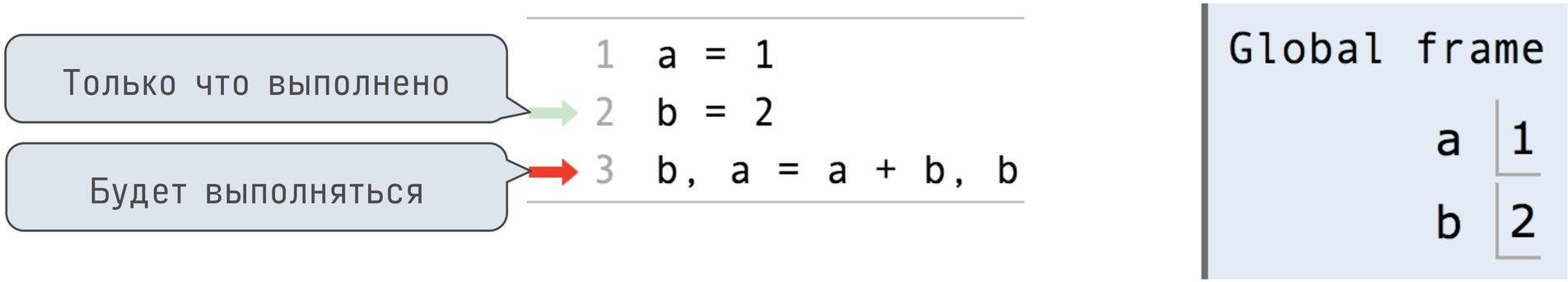


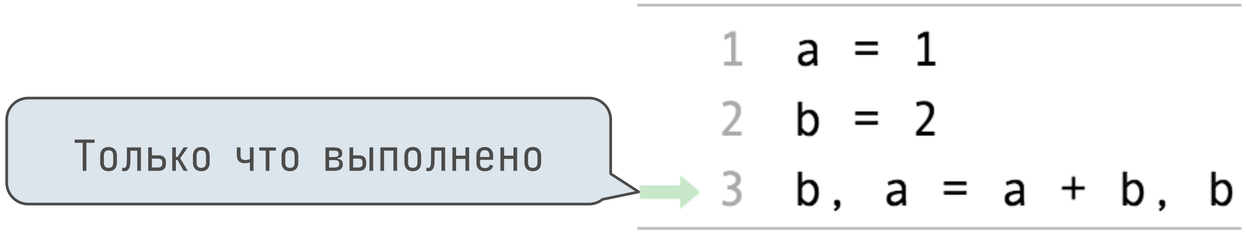
Правила выполнения инструкций присвоения:

1. Вычислить значения всех выражений справа от = слева направо.
2. Связать в текущем фрейме все имена слева от = с полученными значениями.

Интерактивная диаграмма

Инструкция присвоения



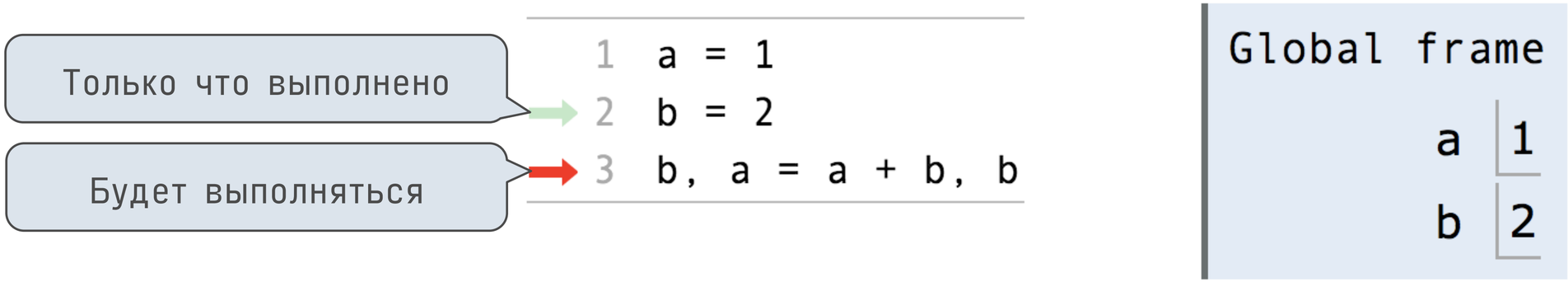


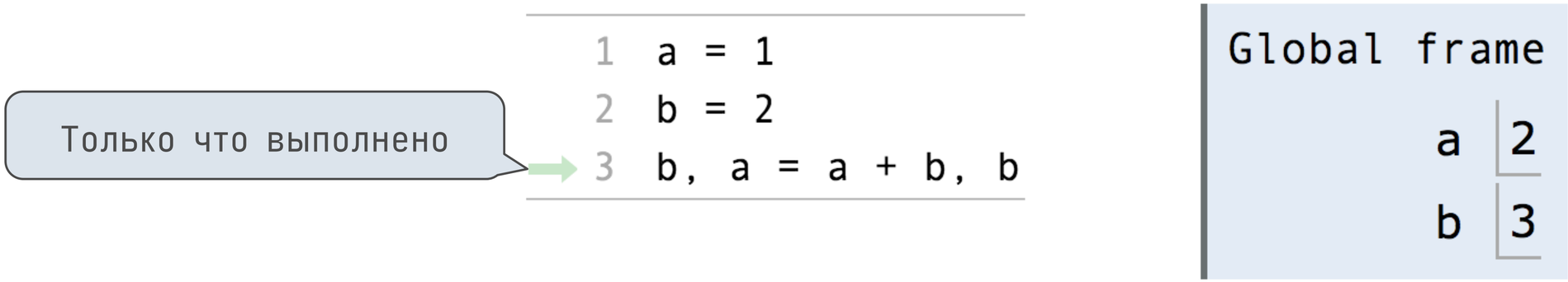
Правила выполнения инструкций присвоения:

1. Вычислить значения всех выражений справа от = слева направо.
2. Связать в текущем фрейме все имена слева от = с полученными значениями.

Интерактивная диаграмма

Инструкция присвоения





Правила выполнения инструкций присвоения:

1. Вычислить значения всех выражений справа от = слева направо.
2. Связать в текущем фрейме все имена слева от = с полученными значениями.

Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

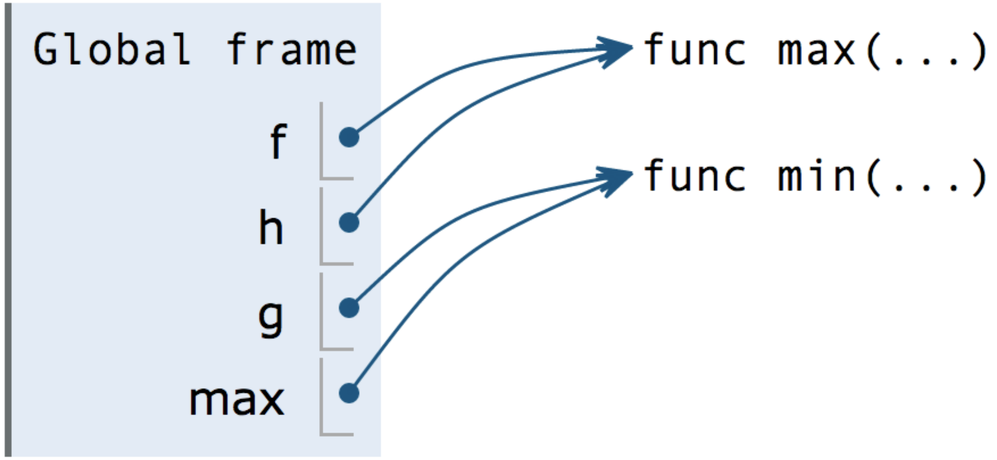
```
1  f = min
2  f = max
3  g, h = min, max
4  max = g
5  max(f(2, g(h(1, 5), 3)), 4)
```

Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

```
1  f = min
2  f = max
3  g, h = min, max
4  max = g
5  max(f(2, g(h(1, 5), 3)), 4)
```

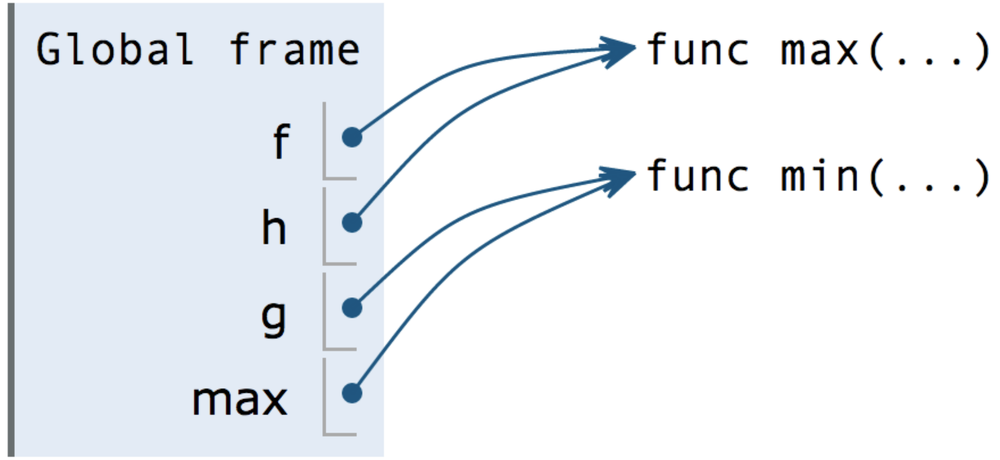


Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

```
1  f = min
2  f = max
3  g, h = min, max
4  max = g
5  max(f(2, g(h(1, 5), 3)), 4)
```



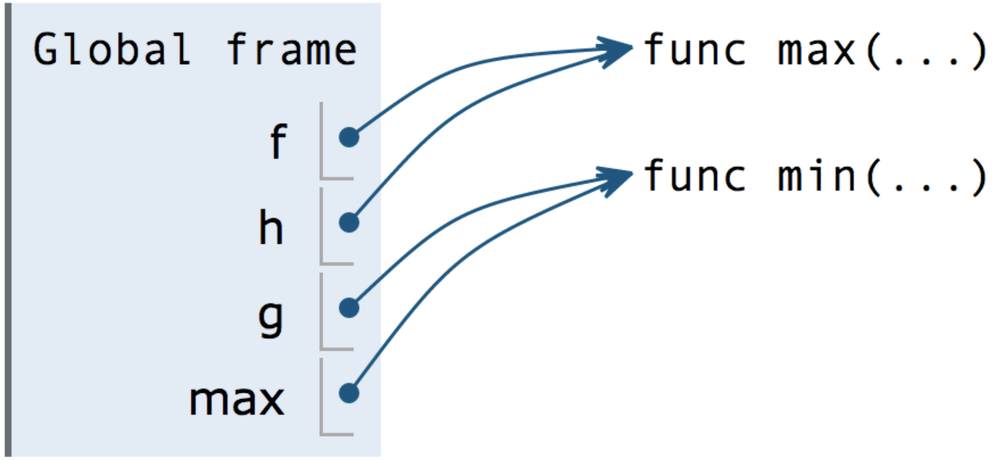
Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

```
1  f = min
2  f = max
3  g, h = min, max
4  max = g
5  max(f(2, g(h(1, 5), 3)), 4)
```

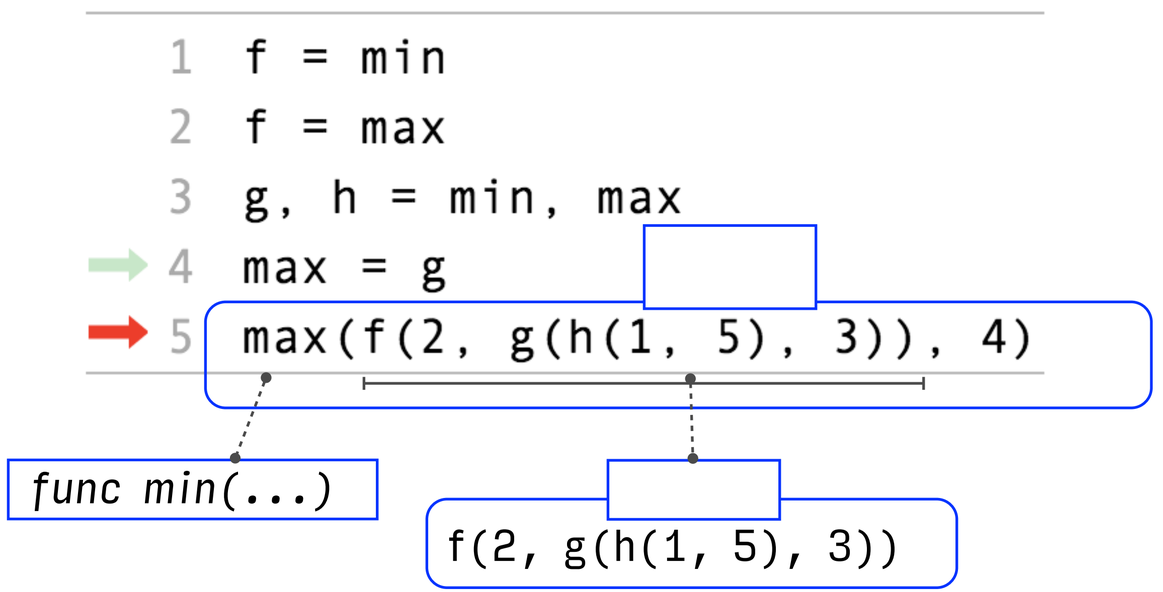
func min(...)

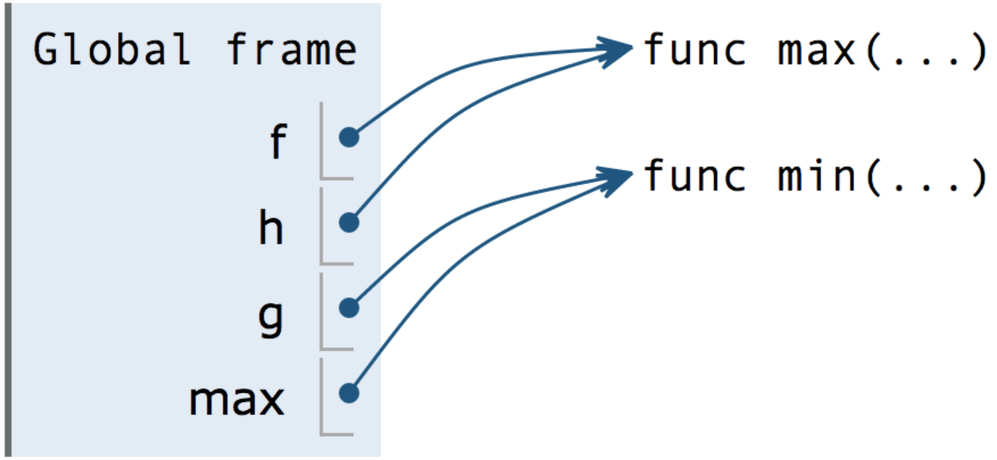


Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

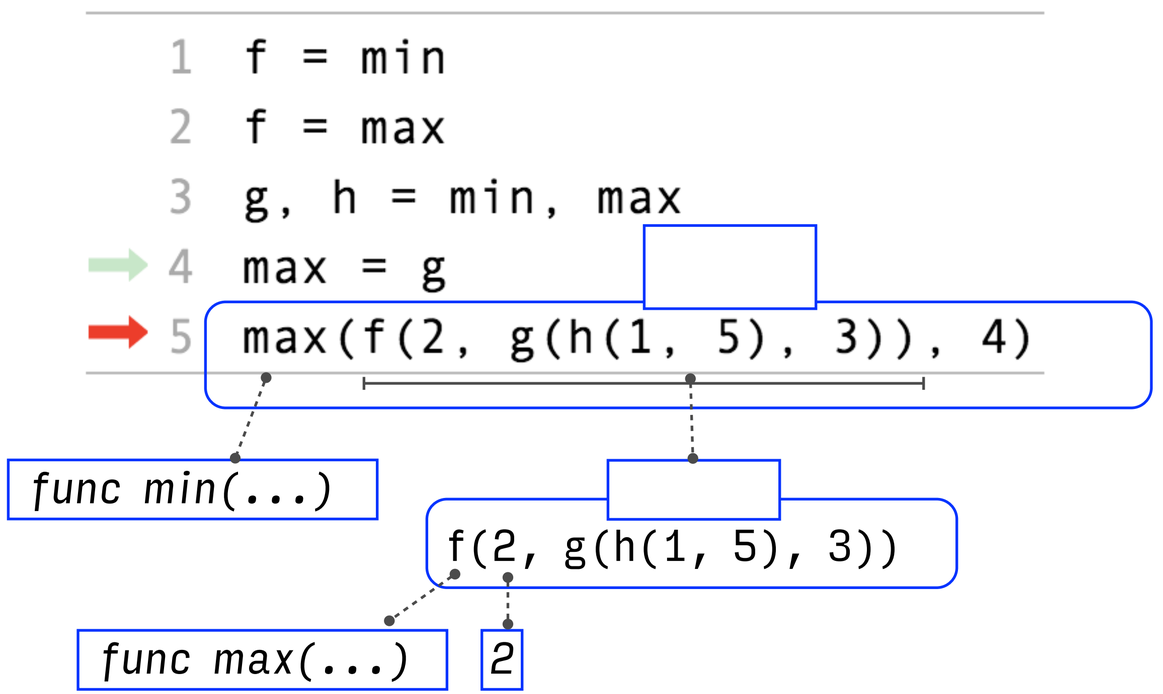


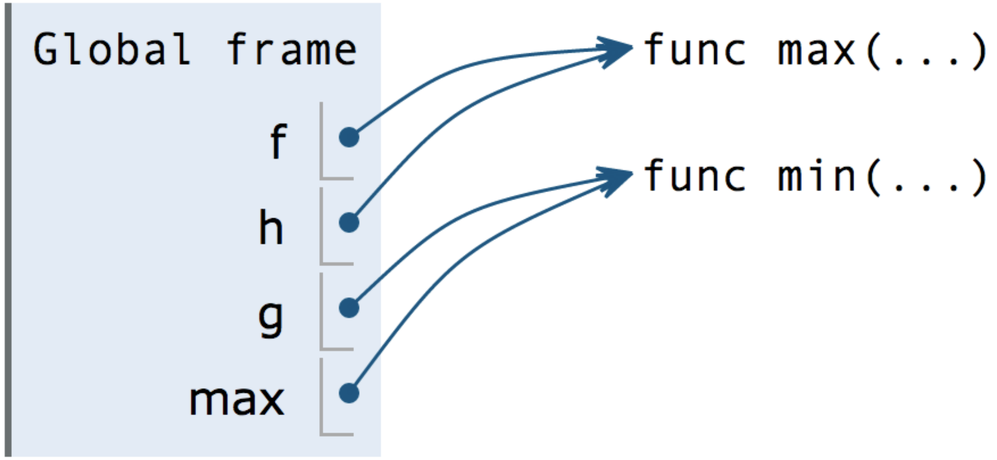


Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

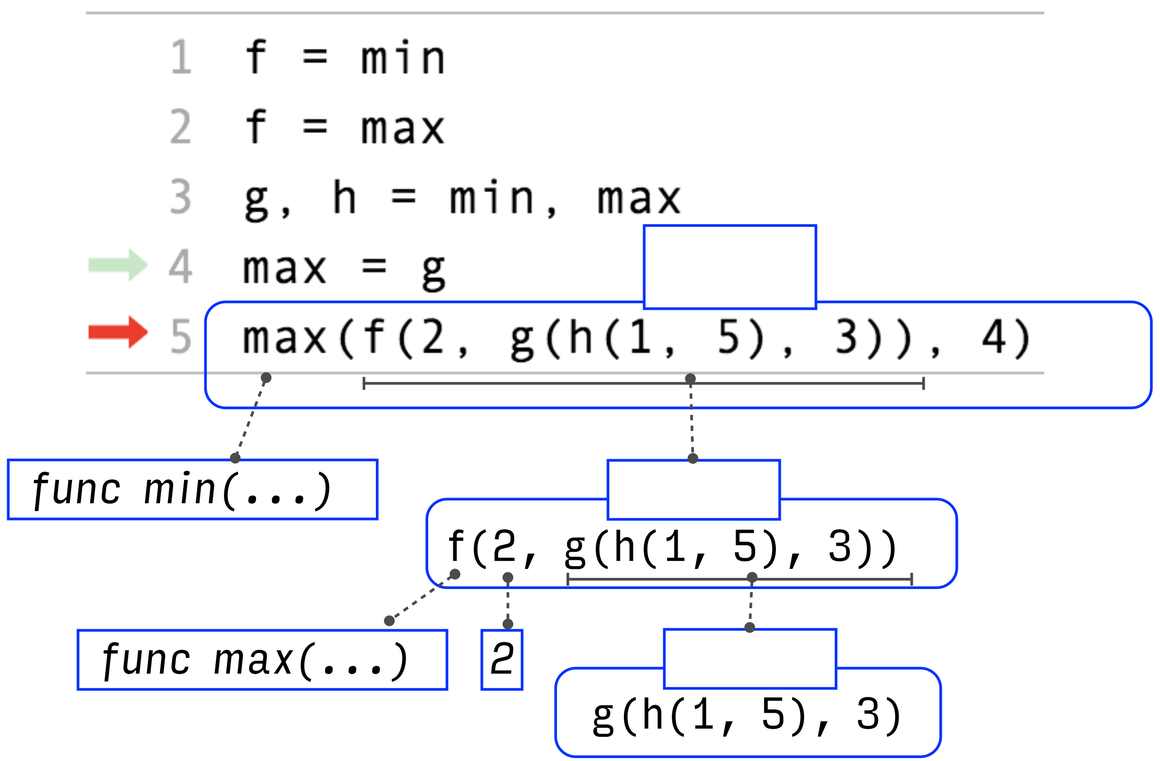


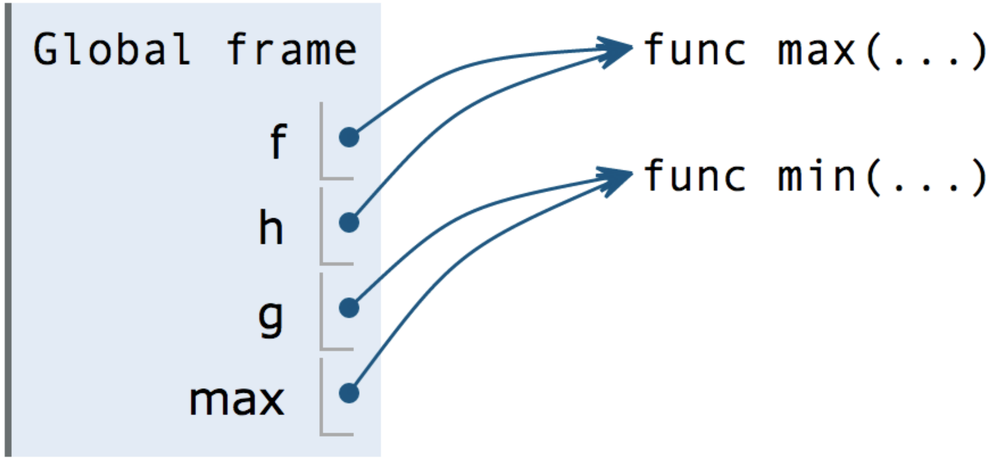


Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

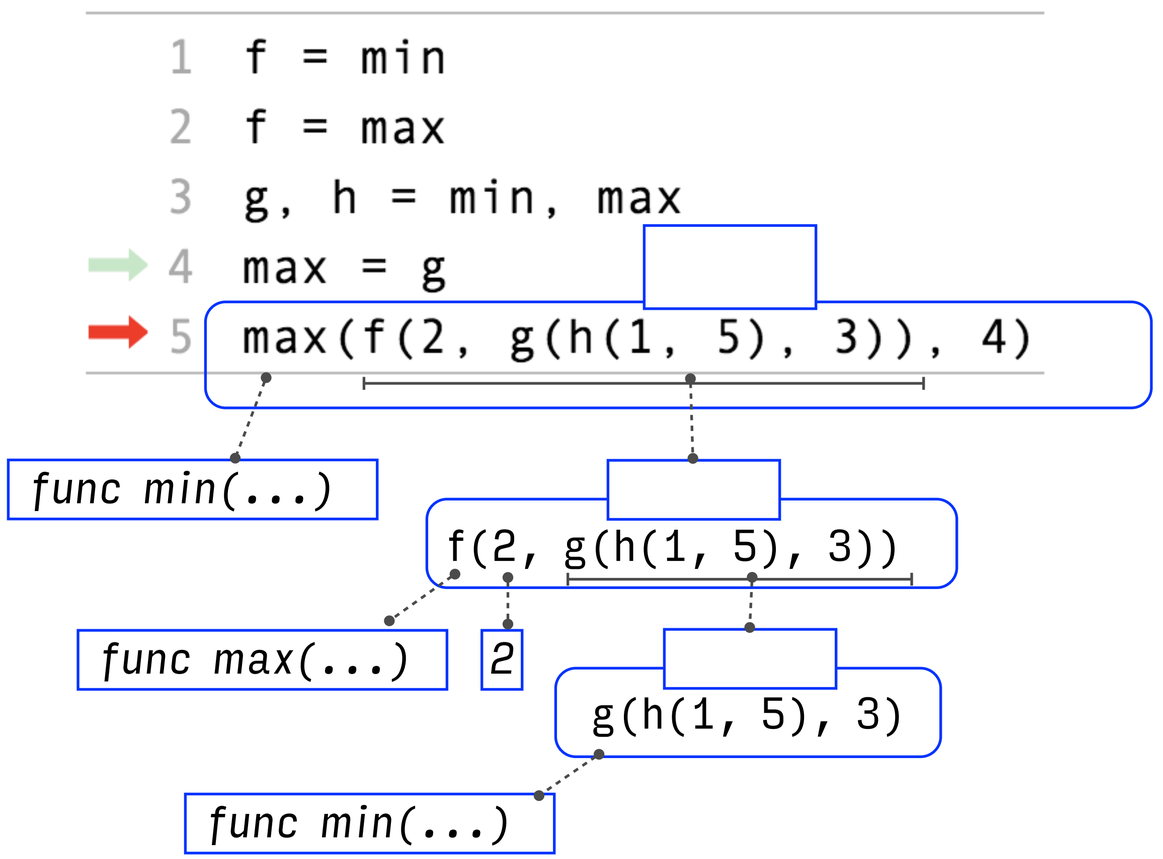


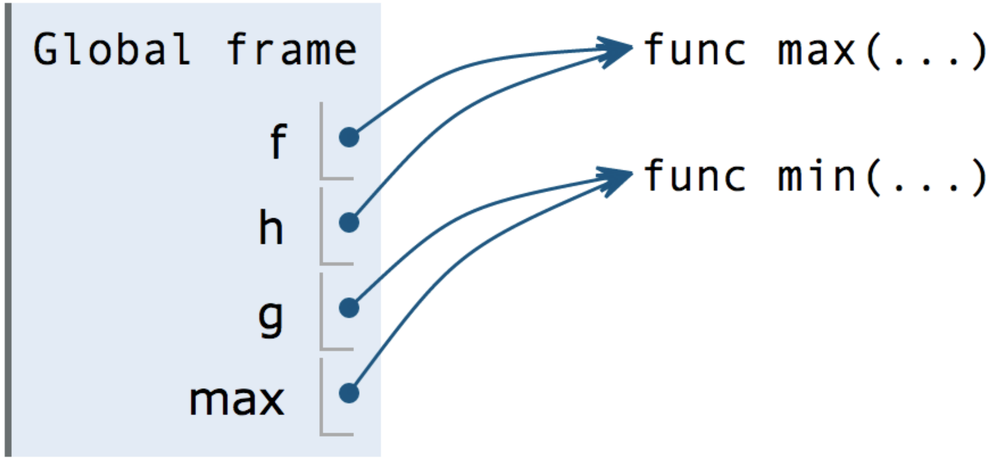


Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

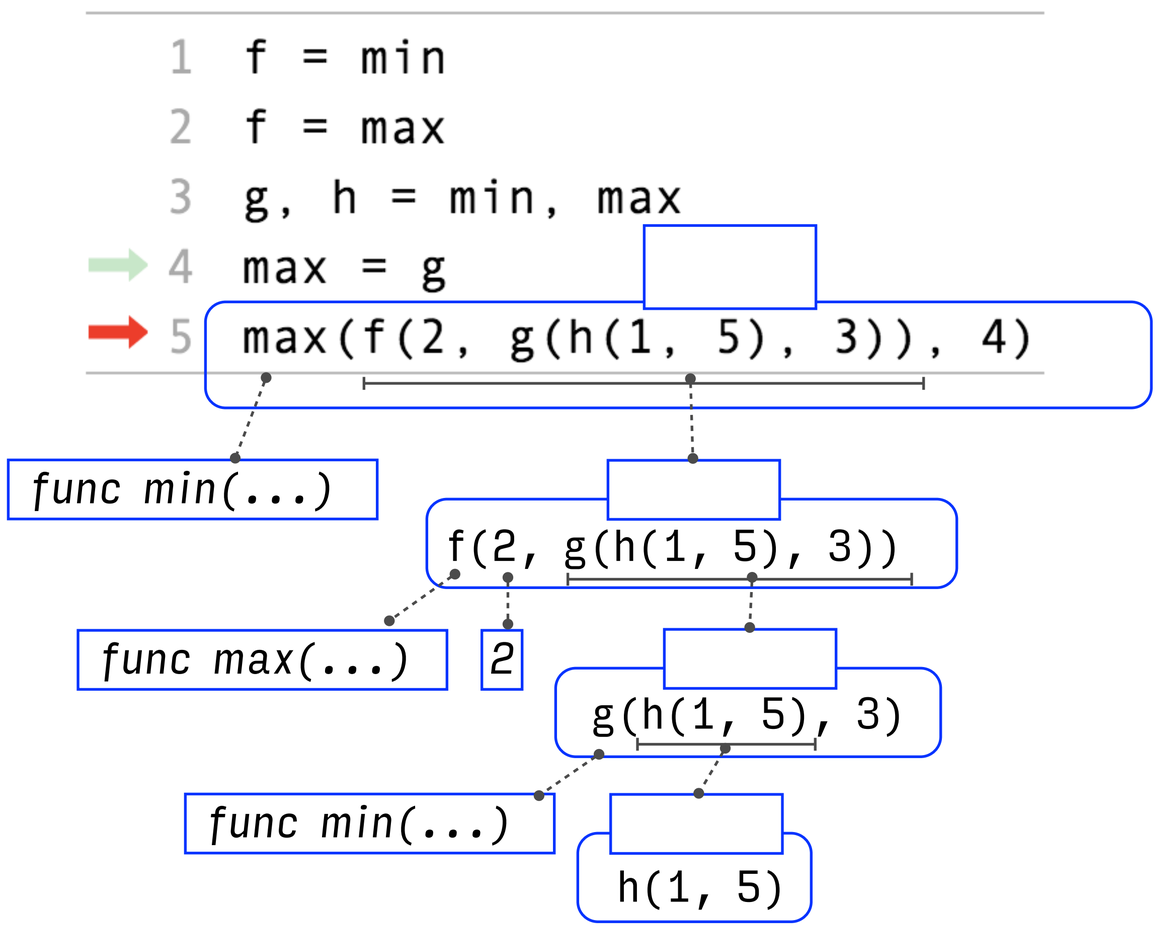


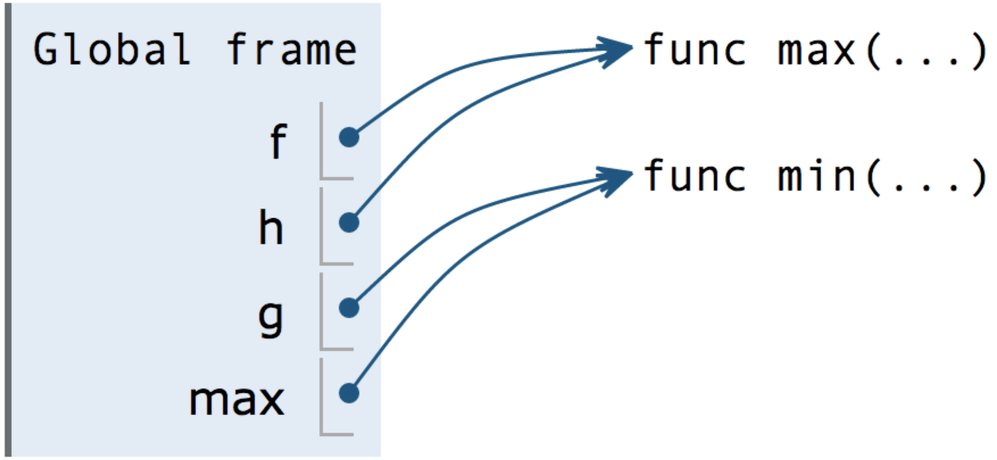


Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

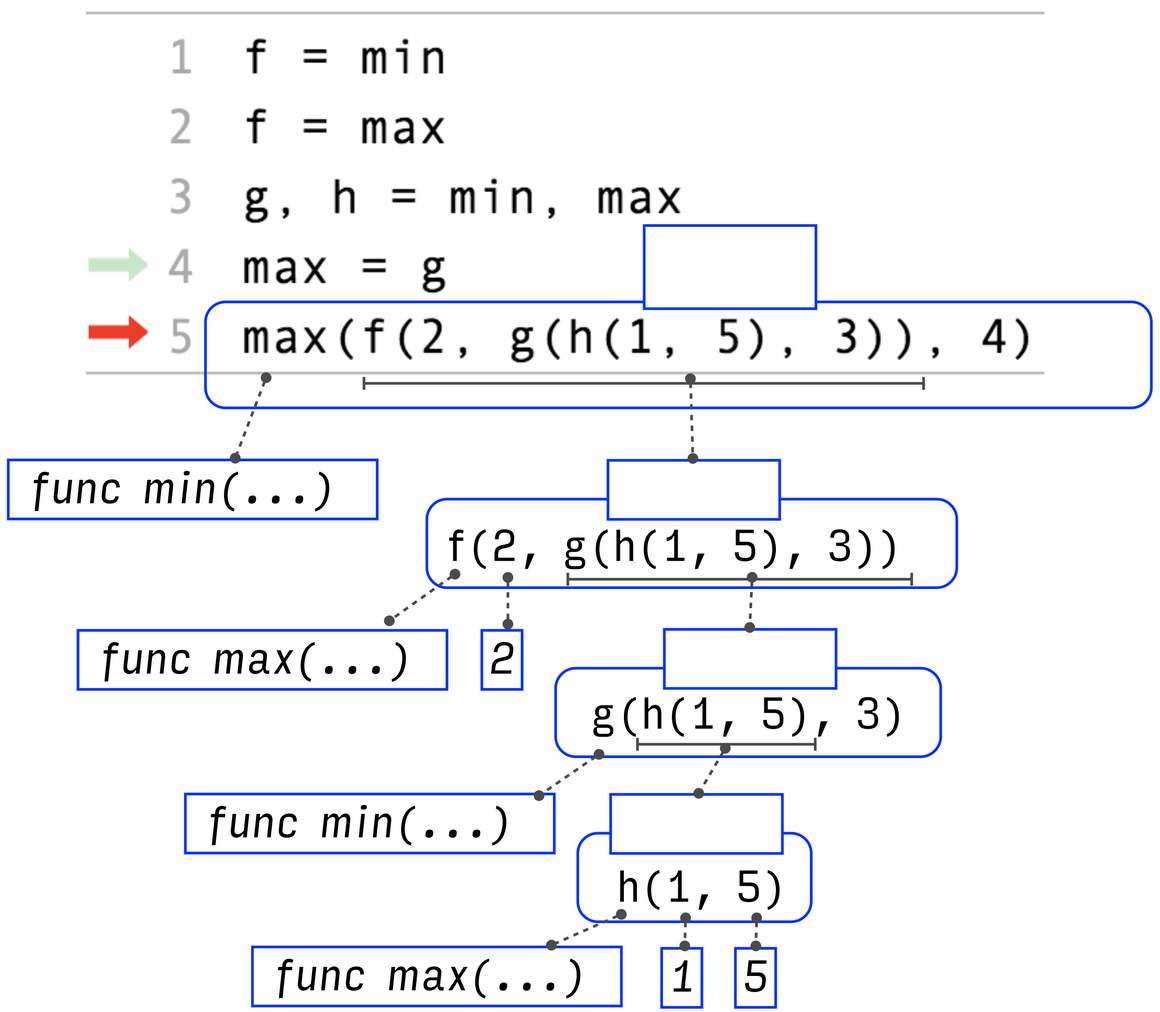


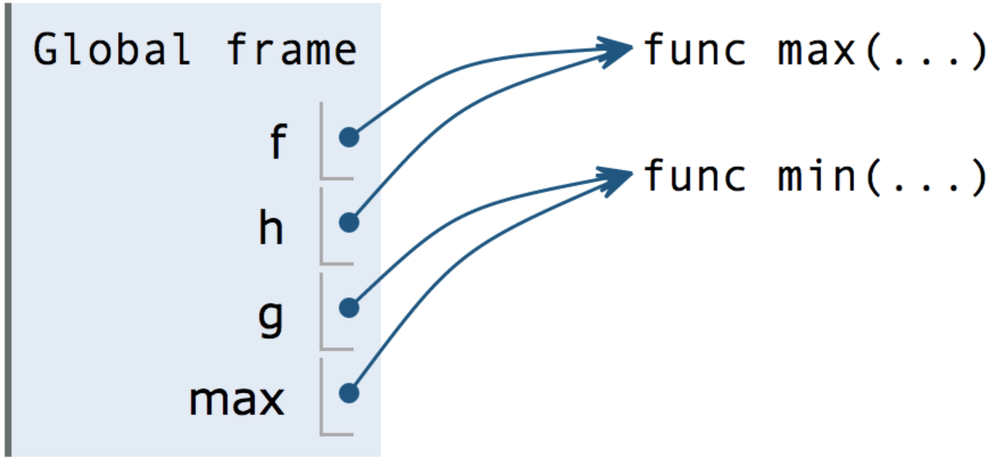


Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

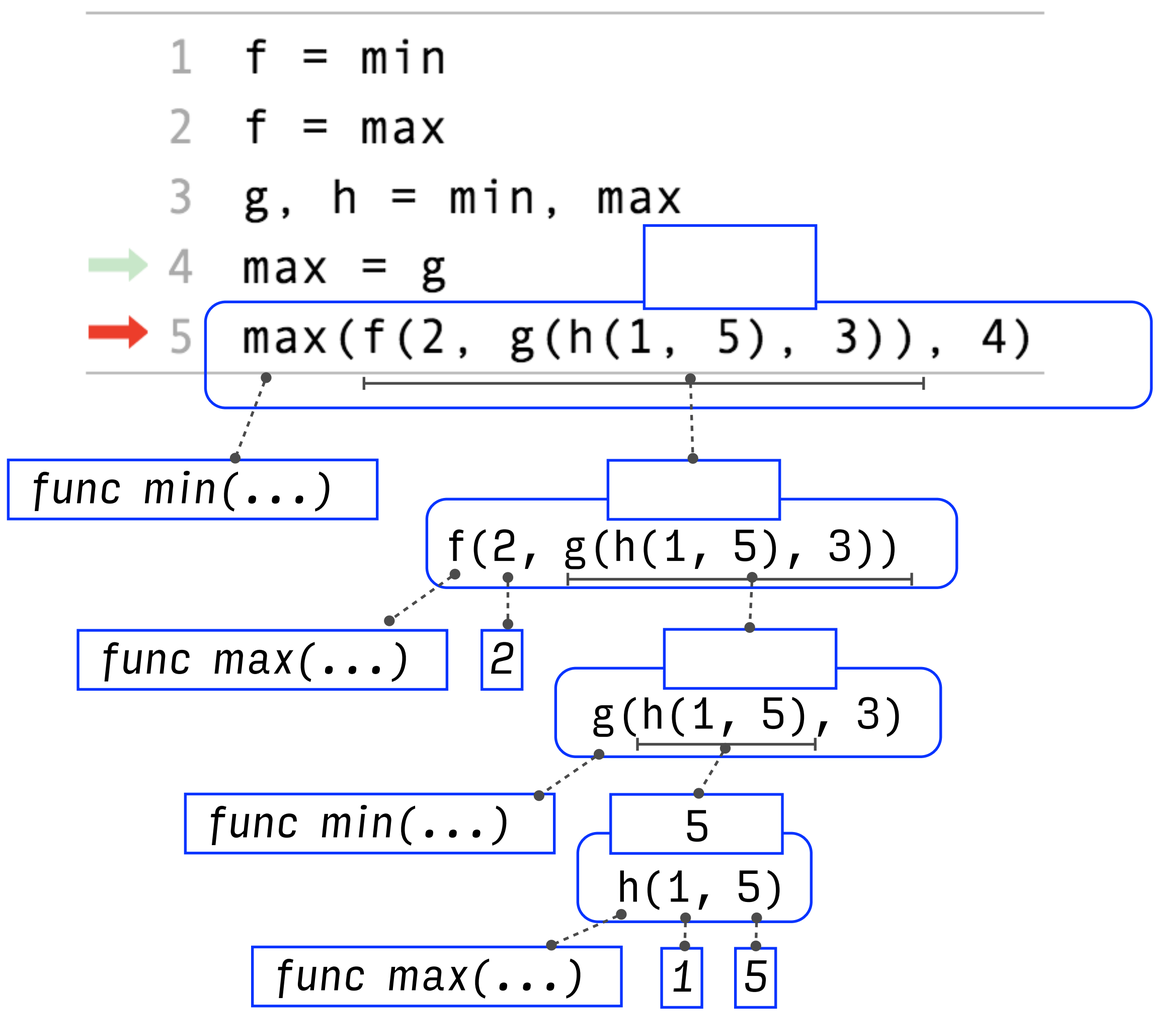


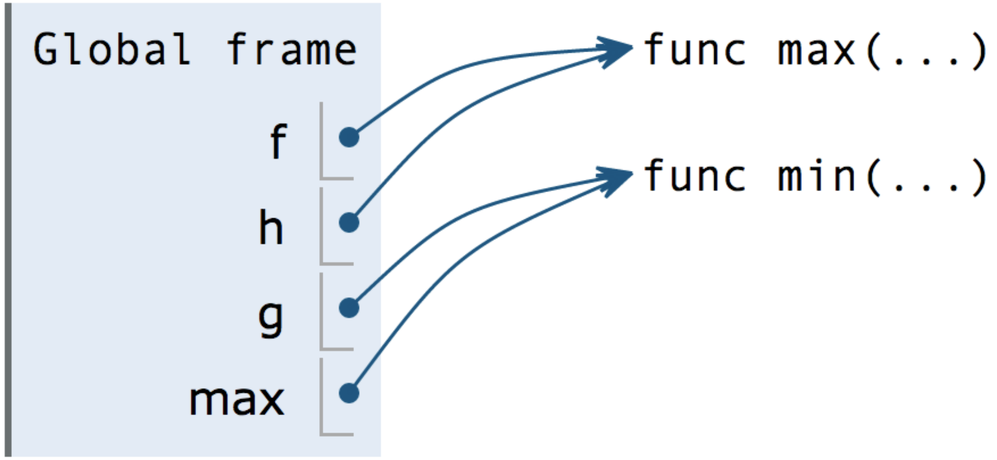


Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

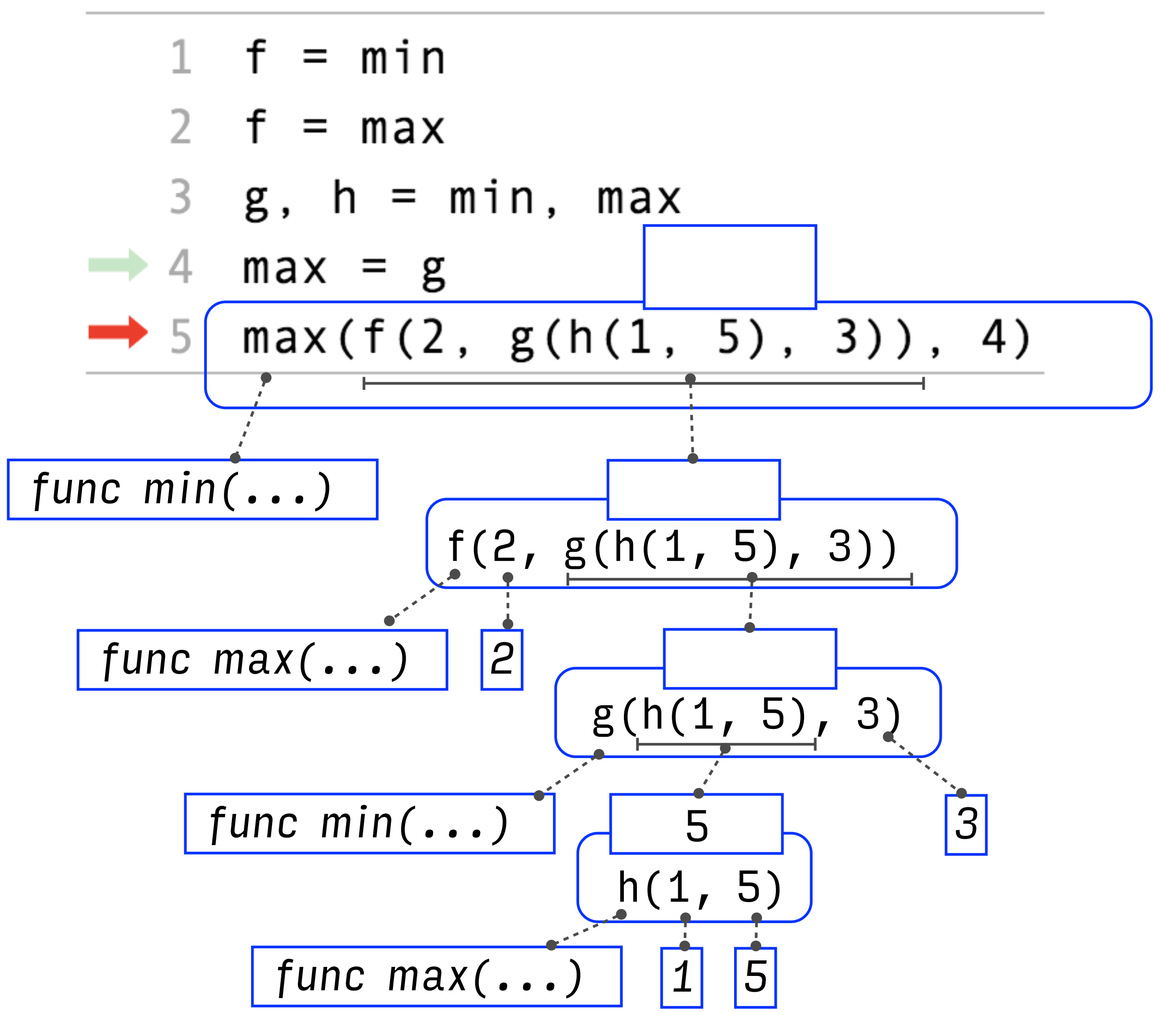


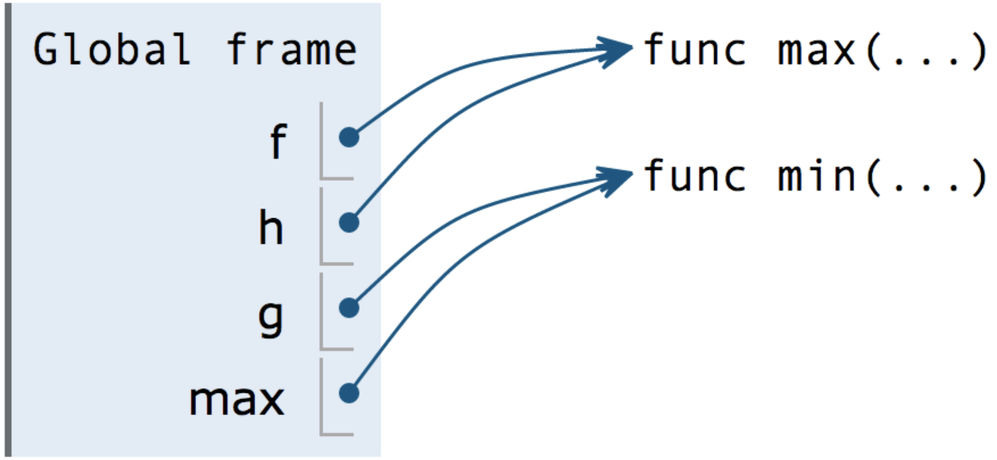


Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

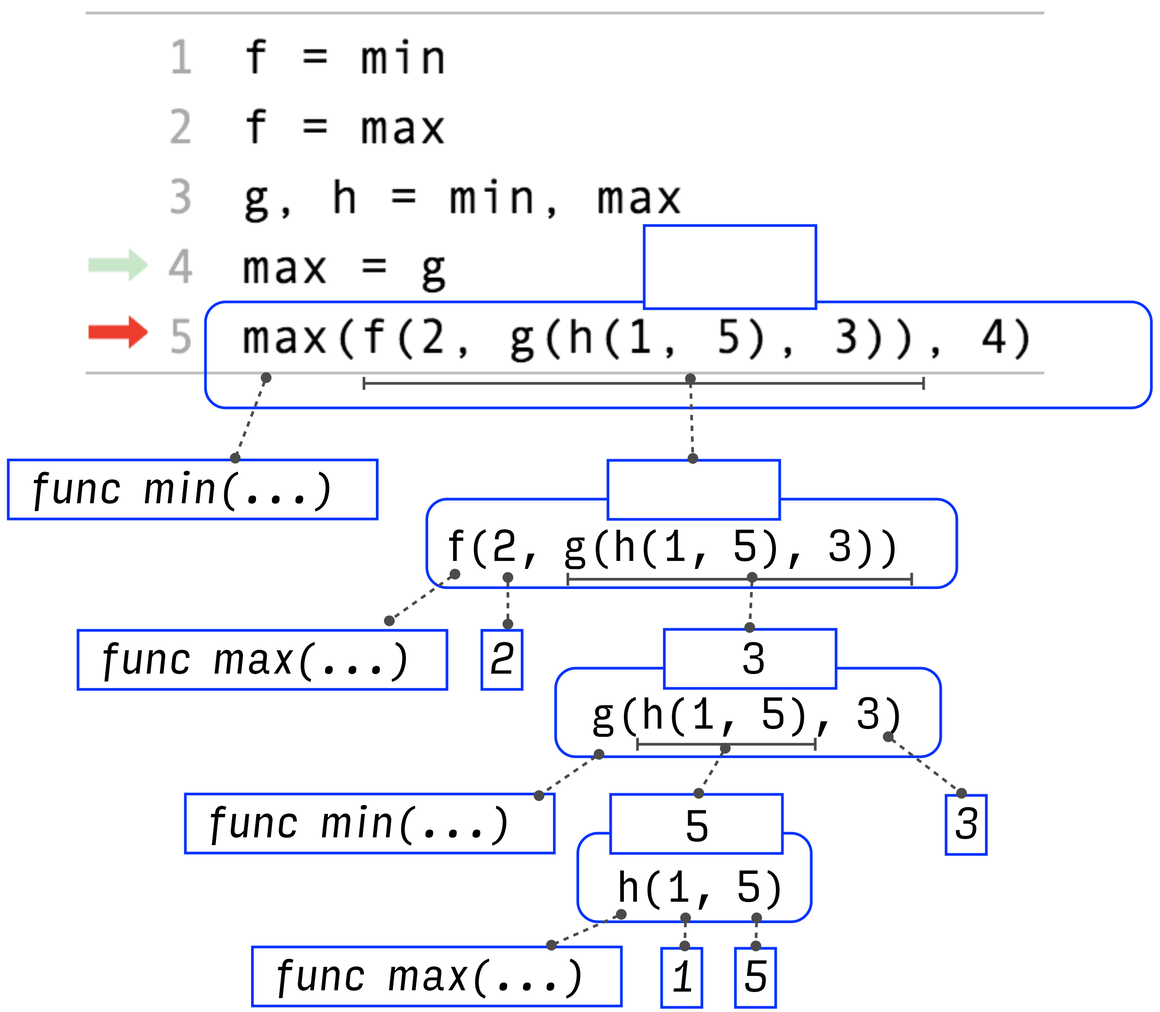


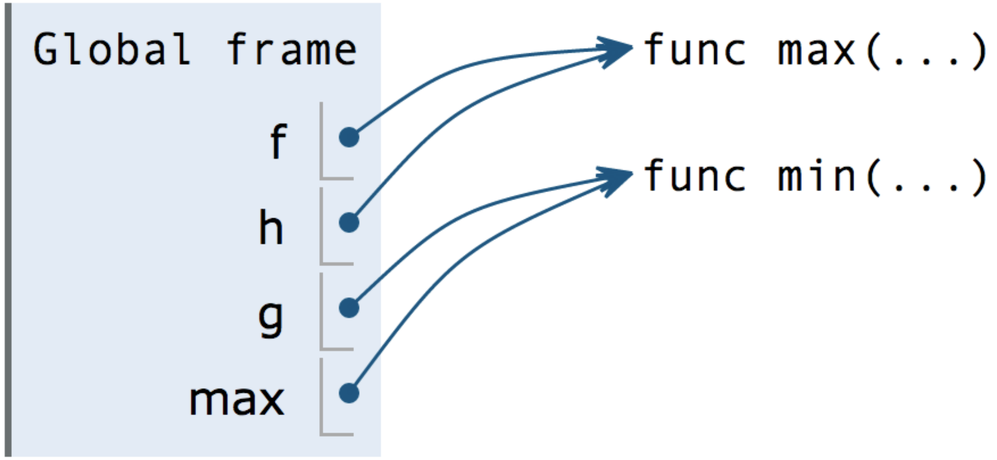


Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

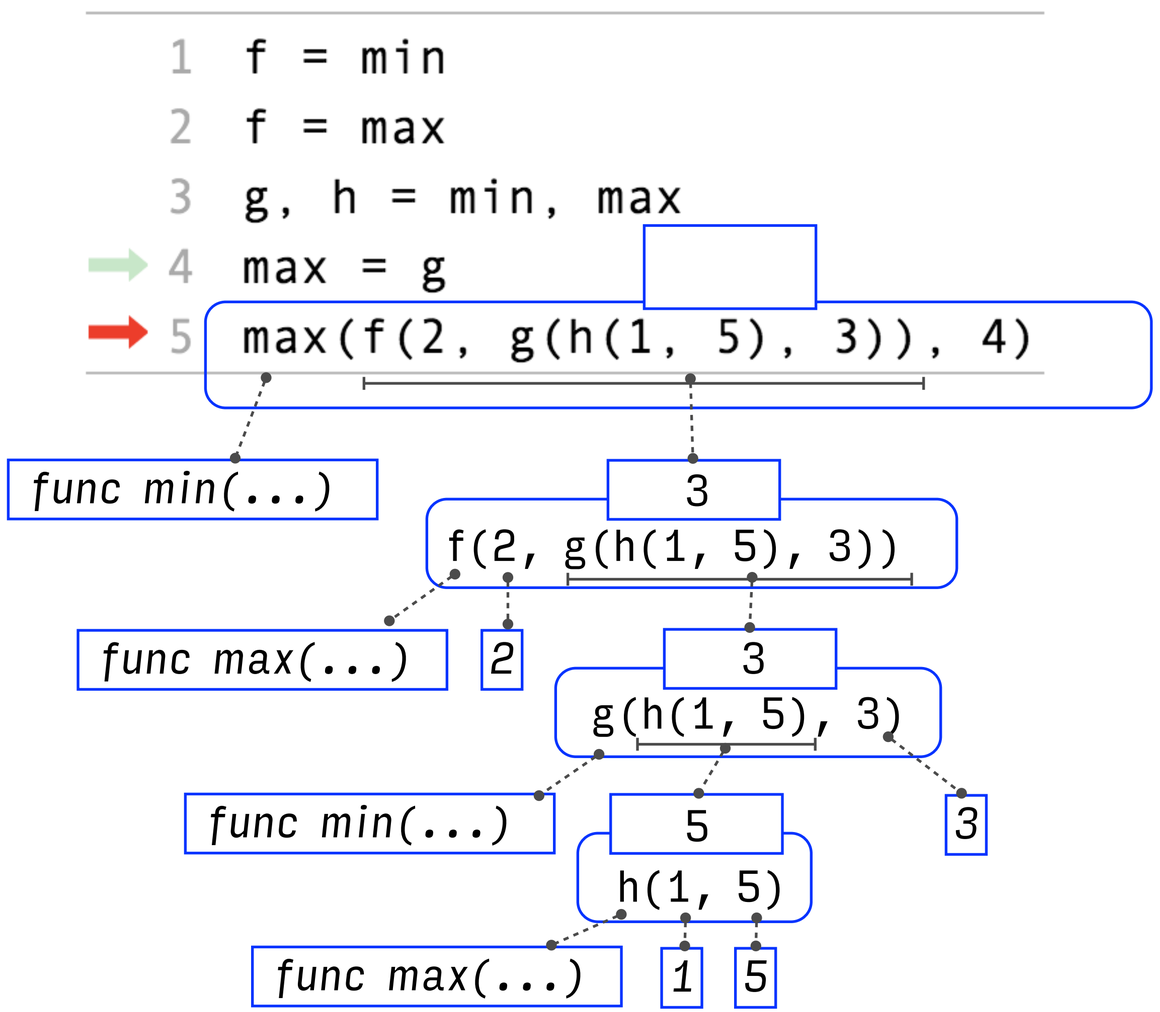


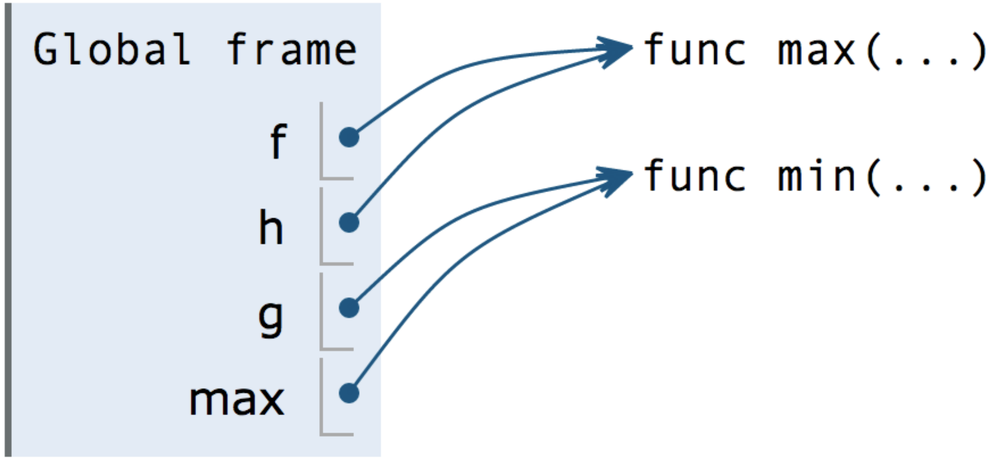


Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

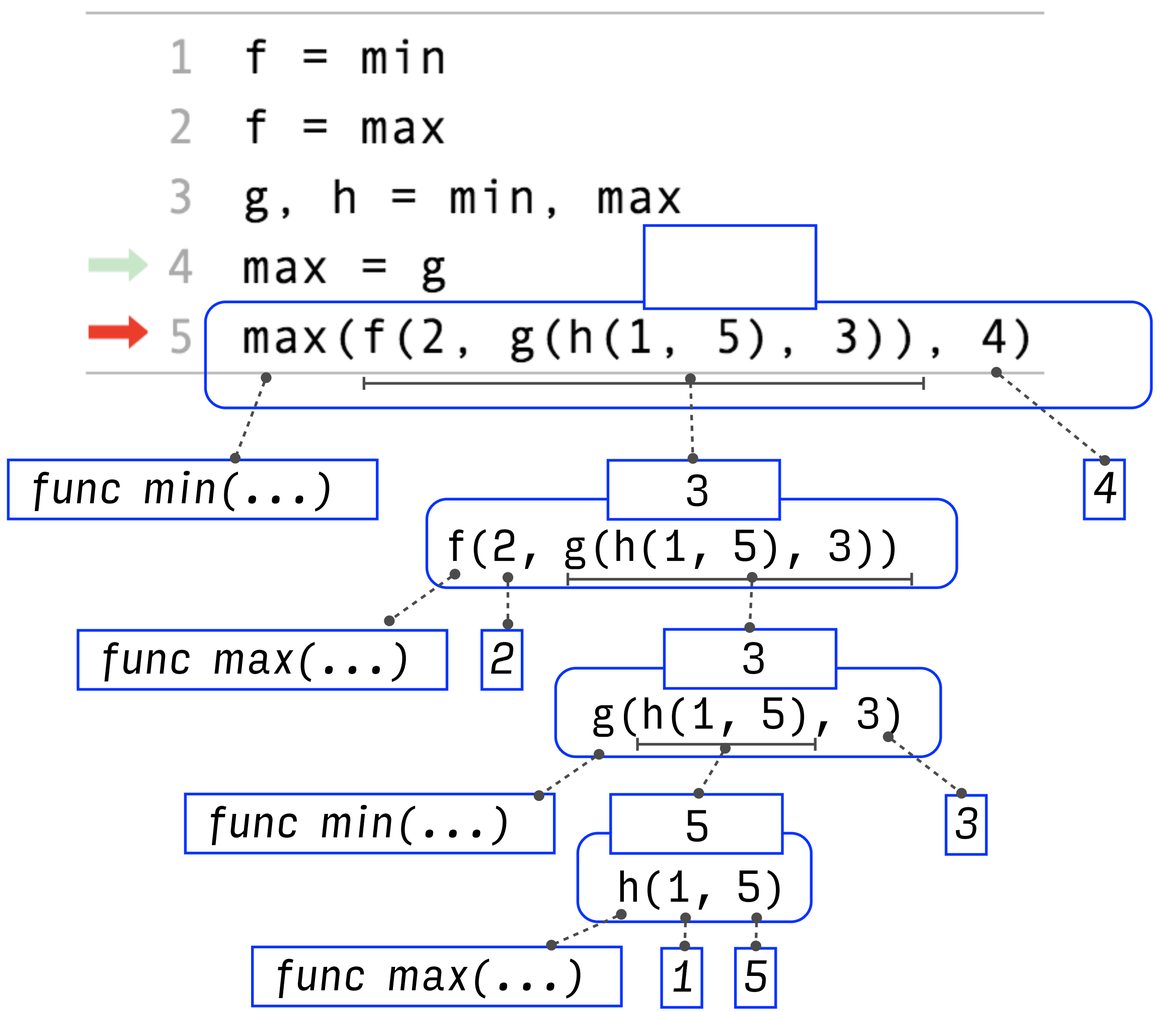


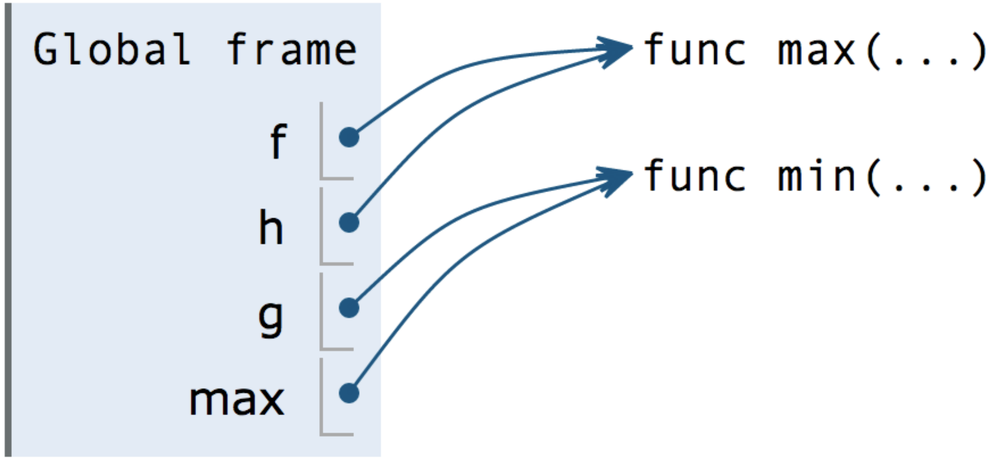


Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

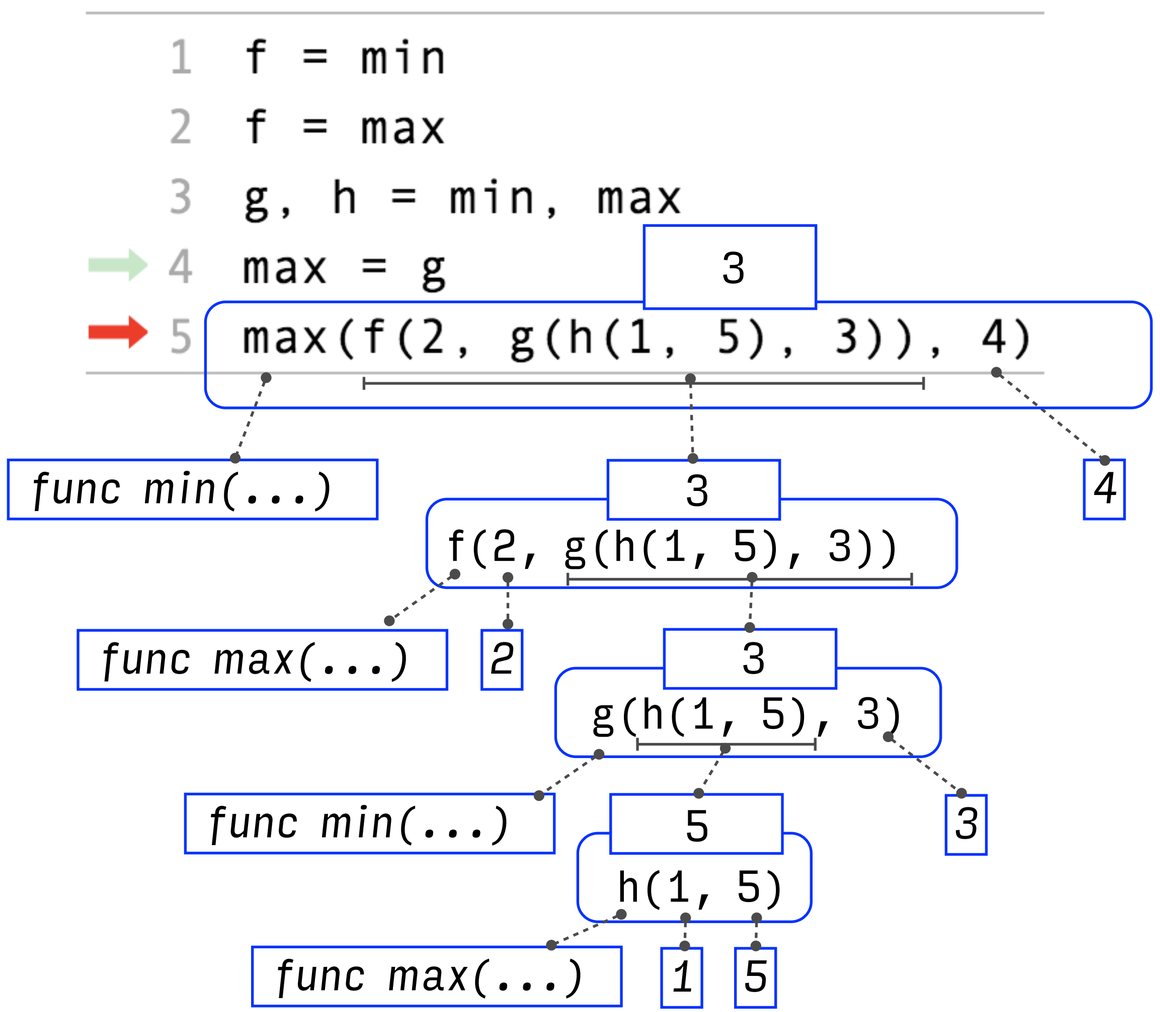


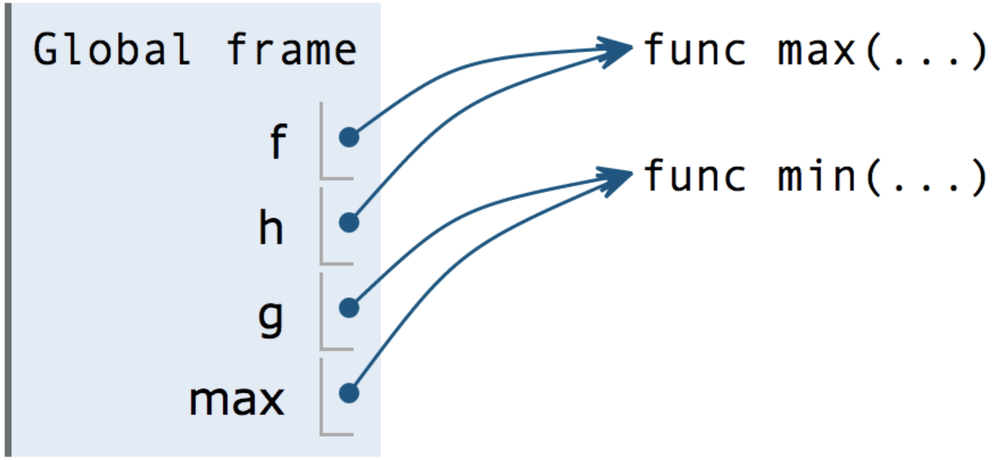


Интерактивная диаграмма

Вопрос 1. Решение

(Пример)

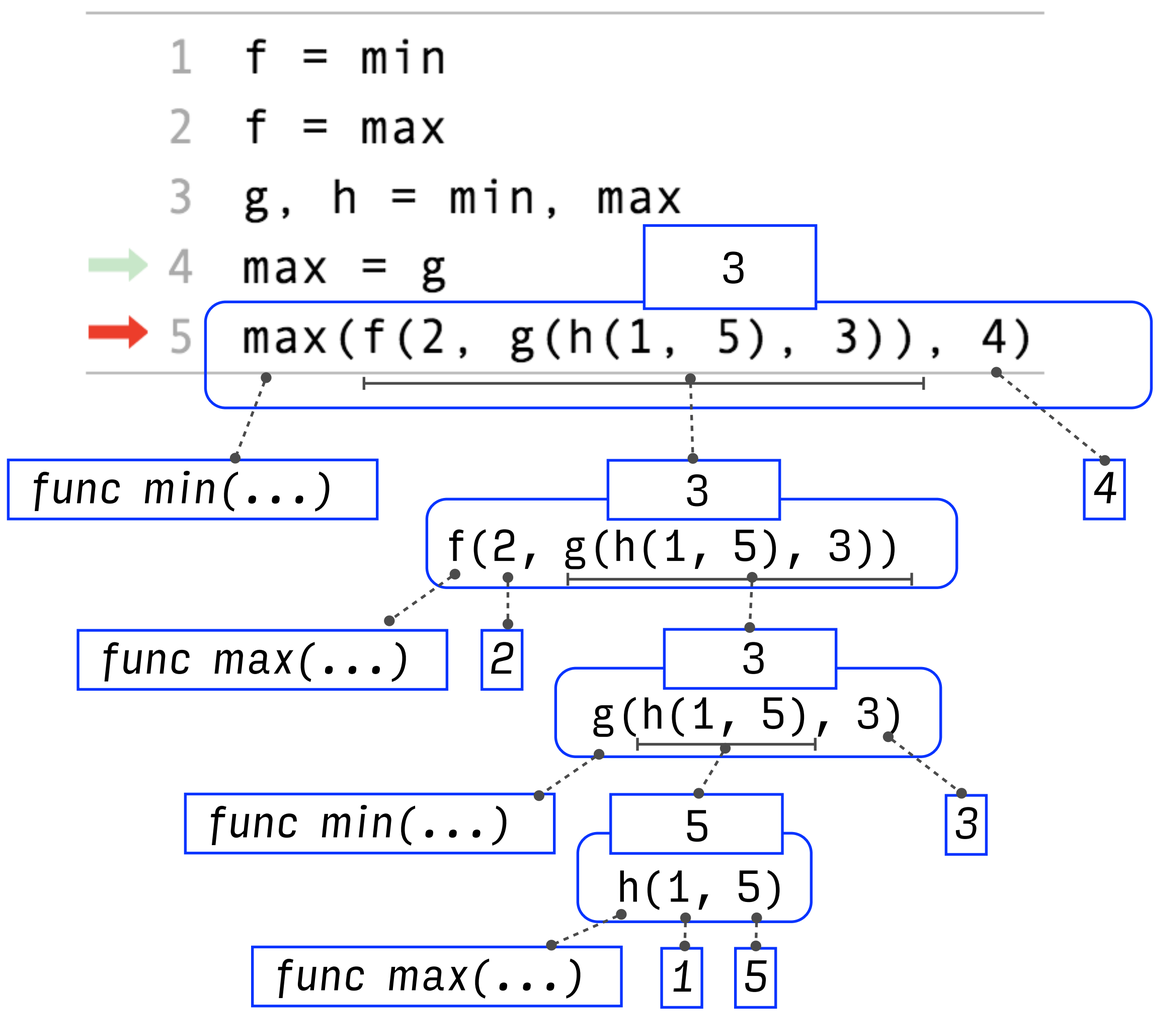


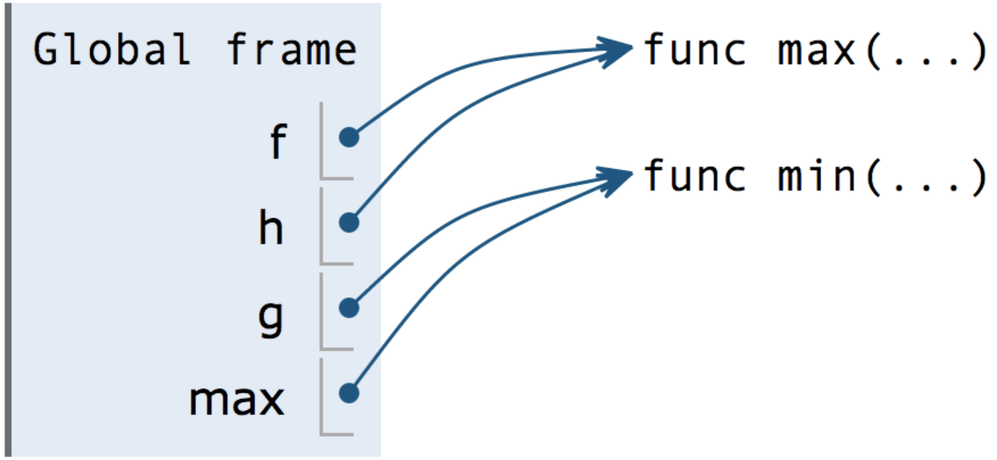


Интерактивная диаграмма

Вопрос 1. Решение

(Пример)





3

Интерактивная диаграмма

Объявление функций

Объявление функций

Присвоение является простым средством абстракции: связывает имена со значениями.

Задание функций является более мощным средством абстракции: связывает имена с выражениями.

Объявление функций

Присвоение является простым средством абстракции: связывает имена со значениями.

Задание функций является более мощным средством абстракции: связывает имена с выражениями.

```
>>> def <имя>(<формальные параметры>):
        return <выражение возврата>
```

Объявление функций

Присвоение является простым средством абстракции: связывает имена со значениями.

Задание функций является более мощным средством абстракции: связывает имена с выражениями.

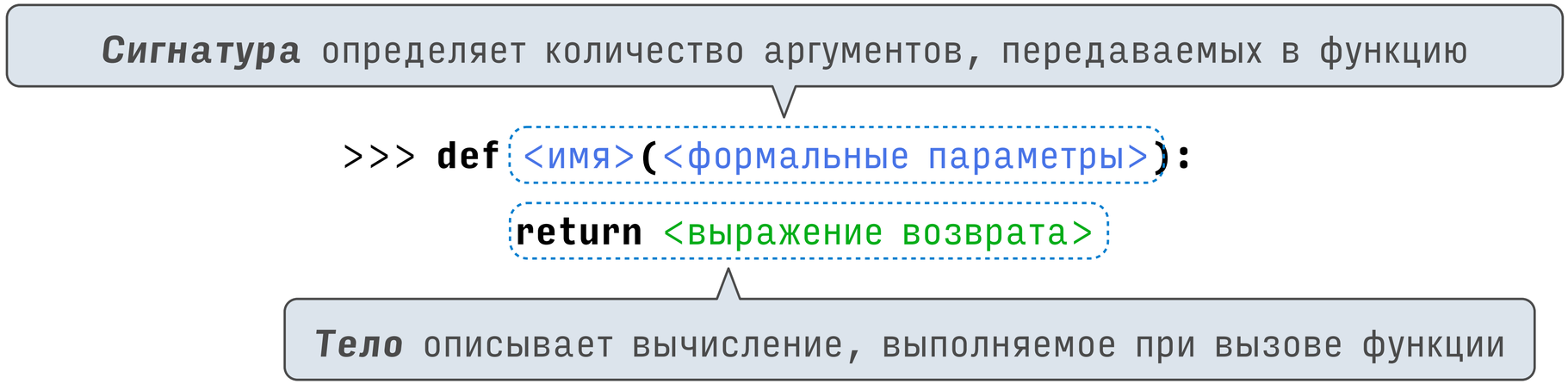
Сигнатура определяет количество аргументов, передаваемых в функцию

```
>>> def <имя>(<формальные параметры>):
        return <выражение возврата>
```

Объявление функций

Присвоение является простым средством абстракции: связывает имена со значениями.

Задание функций является более мощным средством абстракции: связывает имена с выражениями.



Объявление функций

Присвоение является простым средством абстракции: связывает имена со значениями.

Задание функций является более мощным средством абстракции: связывает имена с выражениями.

Сигнатура определяет количество аргументов, передаваемых в функцию

```
>>> def <имя>(<формальные параметры>):
        return <выражение возврата>
```

Тело описывает вычисление, выполняемое при вызове функции

Выполнение инструкции def:

Объявление функций

Присвоение является простым средством абстракции: связывает имена со значениями.

Задание функций является более мощным средством абстракции: связывает имена с выражениями.

Сигнатура определяет количество аргументов, передаваемых в функцию

```
>>> def <имя>(<формальные параметры>):
        return <выражение возврата>
```

Тело описывает вычисление, выполняемое при вызове функции

Выполнение инструкции def:

1. Создать функцию с сигнатурой <имя>(<формальные параметры>).

Объявление функций

Присвоение является простым средством абстракции: связывает имена со значениями.

Задание функций является более мощным средством абстракции: связывает имена с выражениями.

Сигнатура определяет количество аргументов, передаваемых в функцию

```
>>> def <имя>(<формальные параметры>):
        return <выражение возврата>
```

Тело описывает вычисление, выполняемое при вызове функции

Выполнение инструкции def:

1. Создать функцию с сигнатурой <имя>(<формальные параметры>).
2. Определить тело функции по отступам строк.

Объявление функций

Присвоение является простым средством абстракции: связывает имена со значениями.

Задание функций является более мощным средством абстракции: связывает имена с выражениями.

Сигнатура определяет количество аргументов, передаваемых в функцию

```
>>> def <имя>(<формальные параметры>):
        return <выражение возврата>
```

Тело описывает вычисление, выполняемое при вызове функции

Выполнение инструкции def:

1. Создать функцию с сигнатурой <имя>(<формальные параметры>).
2. Определить тело функции по отступам строк.
3. В текущем фрейме связать <имя> с созданной функцией.

Вызов пользовательских функций

Интерактивная диаграмма

Вызов пользовательских функций

Процедура вызова/применения пользовательской функции (версия 1):

Интерактивная диаграмма

Вызов пользовательских функций

Процедура вызова/применения пользовательской функции (версия 1):

1. Создать новый локальный фрейм, сформировав новое окружение.

Интерактивная диаграмма

Вызов пользовательских функций

Процедура вызова/применения пользовательской функции (версия 1):

1. Создать новый локальный фрейм, сформировав новое окружение.
2. Связать в этом фрейме формальные параметры функции с аргументами.

Интерактивная диаграмма

Вызов пользовательских функций

Процедура вызова/применения пользовательской функции (версия 1):

1. Создать новый локальный фрейм, сформировав новое окружение.
2. Связать в этом фрейме формальные параметры функции с аргументами.
3. Выполнить тело функции в новом окружении.

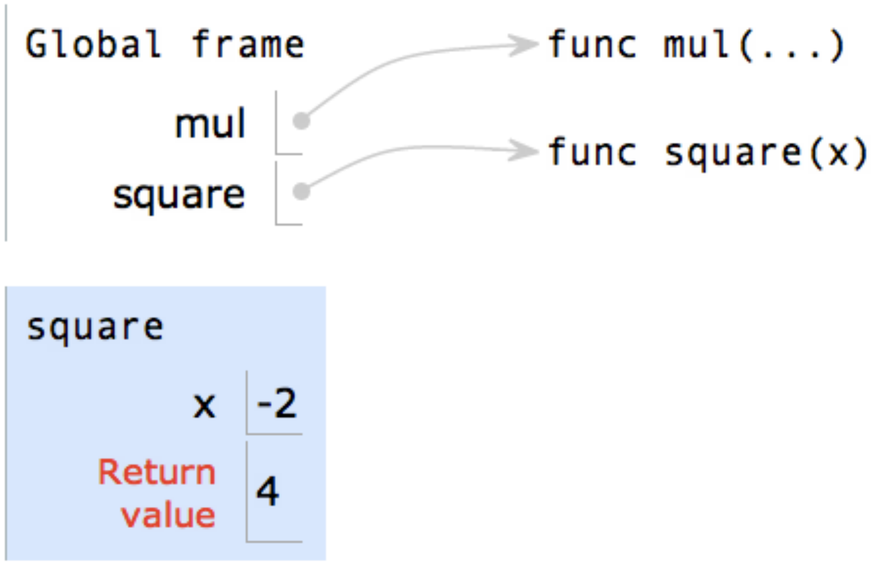
Интерактивная диаграмма

Вызов пользовательских функций

Процедура вызова/применения пользовательской функции (версия 1):

1. Создать новый локальный фрейм, сформировав новое окружение.
2. Связать в этом фрейме формальные параметры функции с аргументами.
3. Выполнить тело функции в новом окружении.

```
1 from operator import mul
2 def square(x):
3     return mul(x, x)
4 square(-2)
```



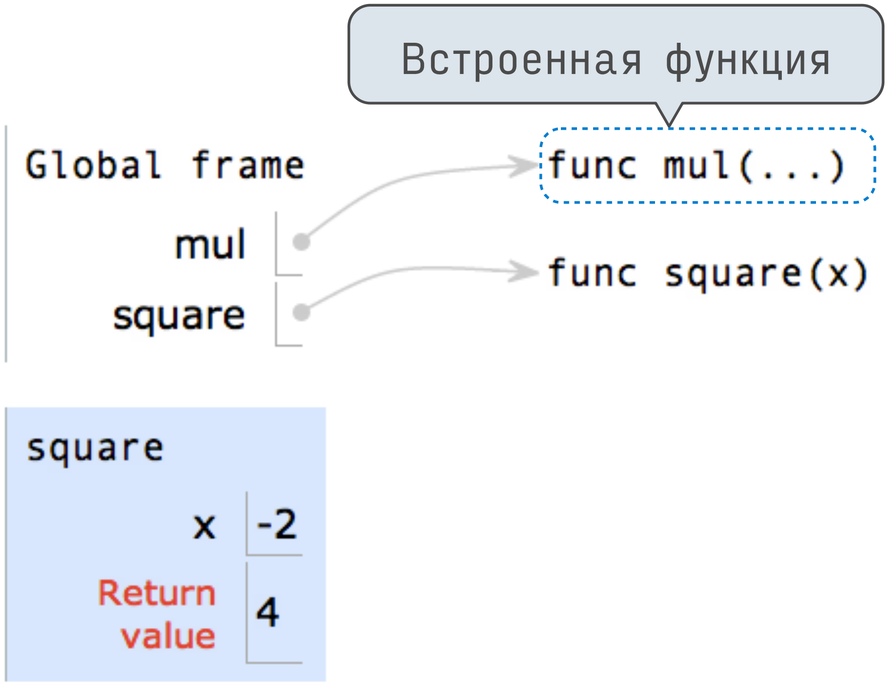
Интерактивная диаграмма

Вызов пользовательских функций

Процедура вызова/применения пользовательской функции (версия 1):

1. Создать новый локальный фрейм, сформировав новое окружение.
2. Связать в этом фрейме формальные параметры функции с аргументами.
3. Выполнить тело функции в новом окружении.

```
1 from operator import mul
2 def square(x):
3     return mul(x, x)
4 square(-2)
```



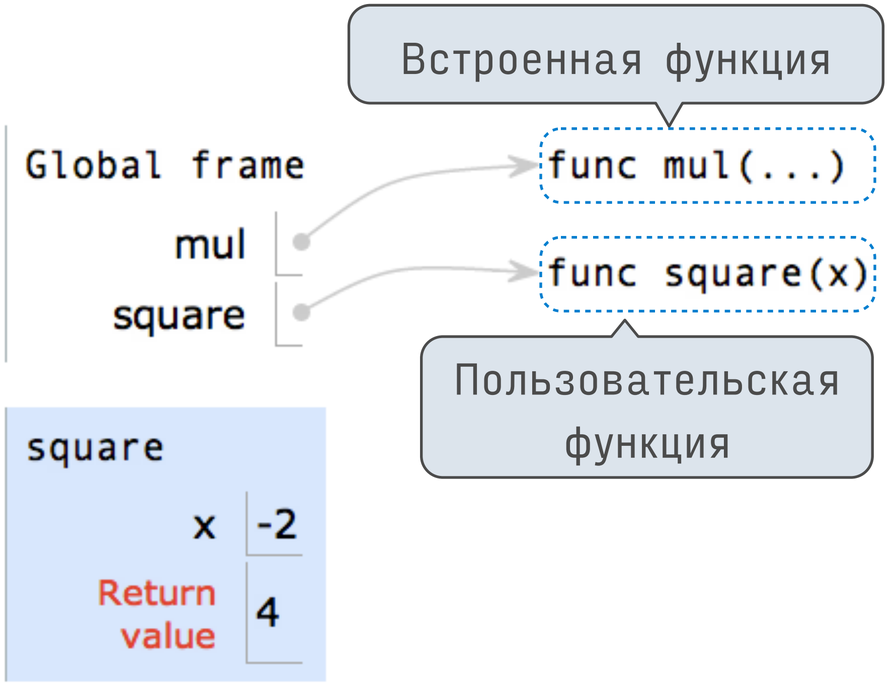
Интерактивная диаграмма

Вызов пользовательских функций

Процедура вызова/применения пользовательской функции (версия 1):

1. Создать новый локальный фрейм, сформировав новое окружение.
2. Связать в этом фрейме формальные параметры функции с аргументами.
3. Выполнить тело функции в новом окружении.

```
1 from operator import mul
2 def square(x):
3     return mul(x, x)
4 square(-2)
```



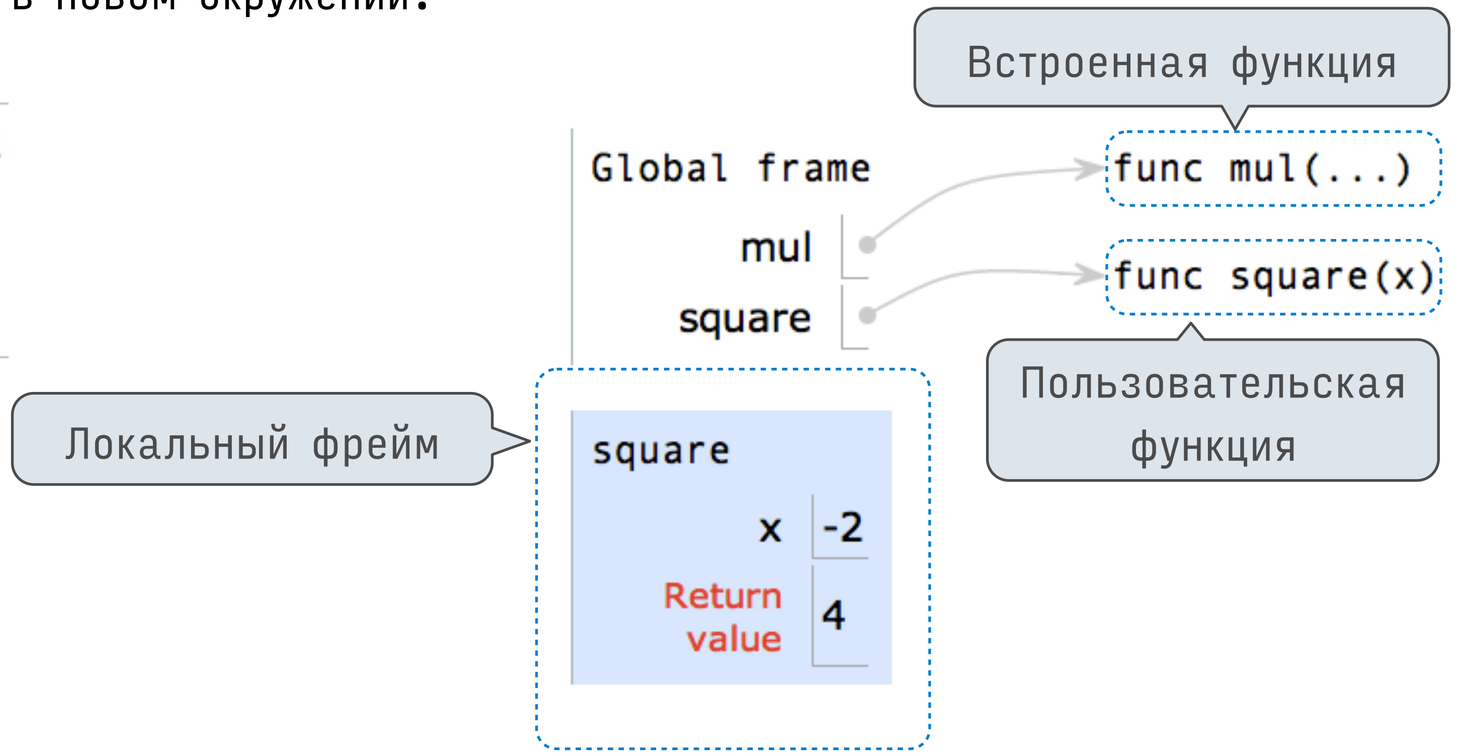
Интерактивная диаграмма

Вызов пользовательских функций

Процедура вызова/применения пользовательской функции (версия 1):

1. Создать новый локальный фрейм, сформировав новое окружение.
2. Связать в этом фрейме формальные параметры функции с аргументами.
3. Выполнить тело функции в новом окружении.

```
1  from operator import mul
2  def square(x):
3      return mul(x, x)
4  square(-2)
```



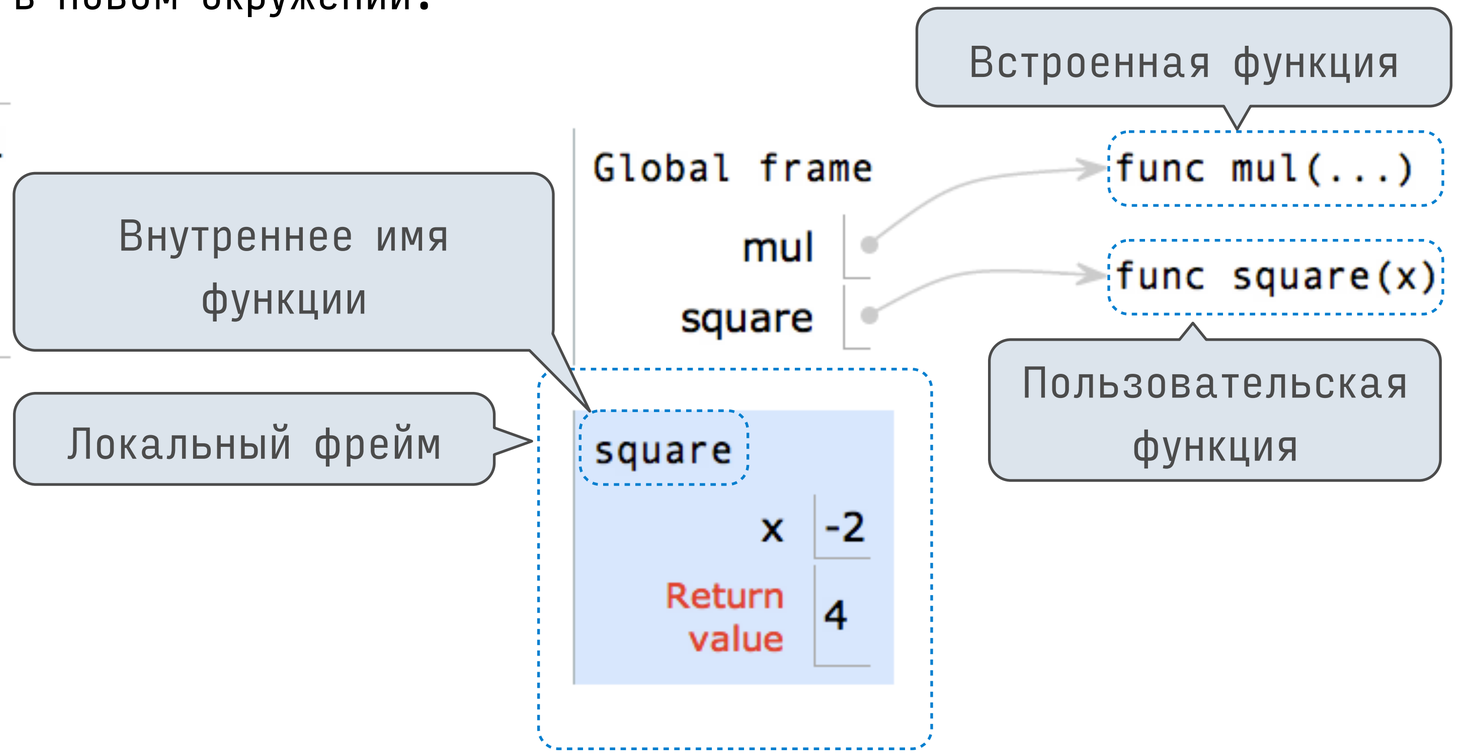
Интерактивная диаграмма

Вызов пользовательских функций

Процедура вызова/применения пользовательской функции (версия 1):

1. Создать новый локальный фрейм, сформировав новое окружение.
2. Связать в этом фрейме формальные параметры функции с аргументами.
3. Выполнить тело функции в новом окружении.

```
1  from operator import mul
2  def square(x):
3      return mul(x, x)
4  square(-2)
```



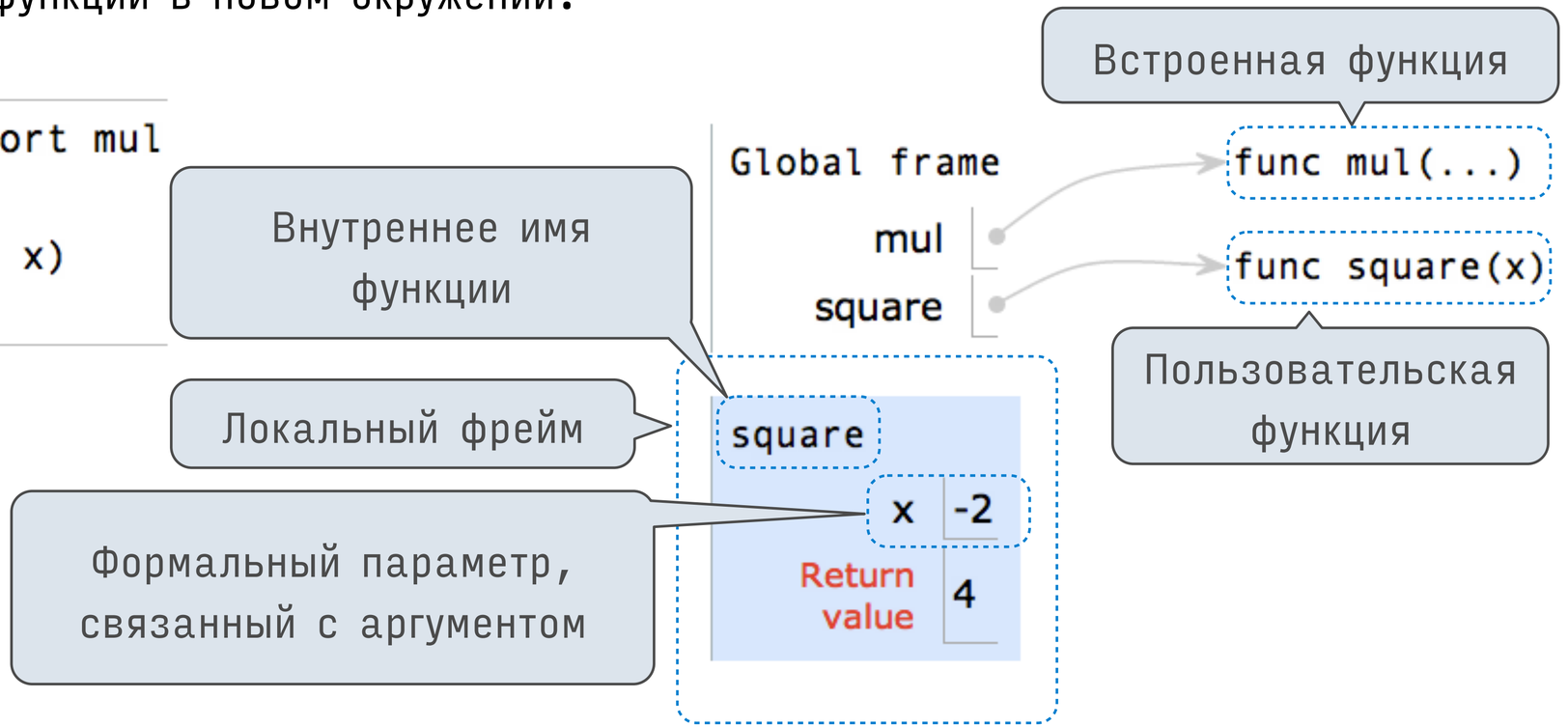
Интерактивная диаграмма

Вызов пользовательских функций

Процедура вызова/применения пользовательской функции (версия 1):

1. Создать новый локальный фрейм, сформировав новое окружение.
2. Связать в этом фрейме формальные параметры функции с аргументами.
3. Выполнить тело функции в новом окружении.

```
1  from operator import mul
2  def square(x):
3      return mul(x, x)
4  square(-2)
```



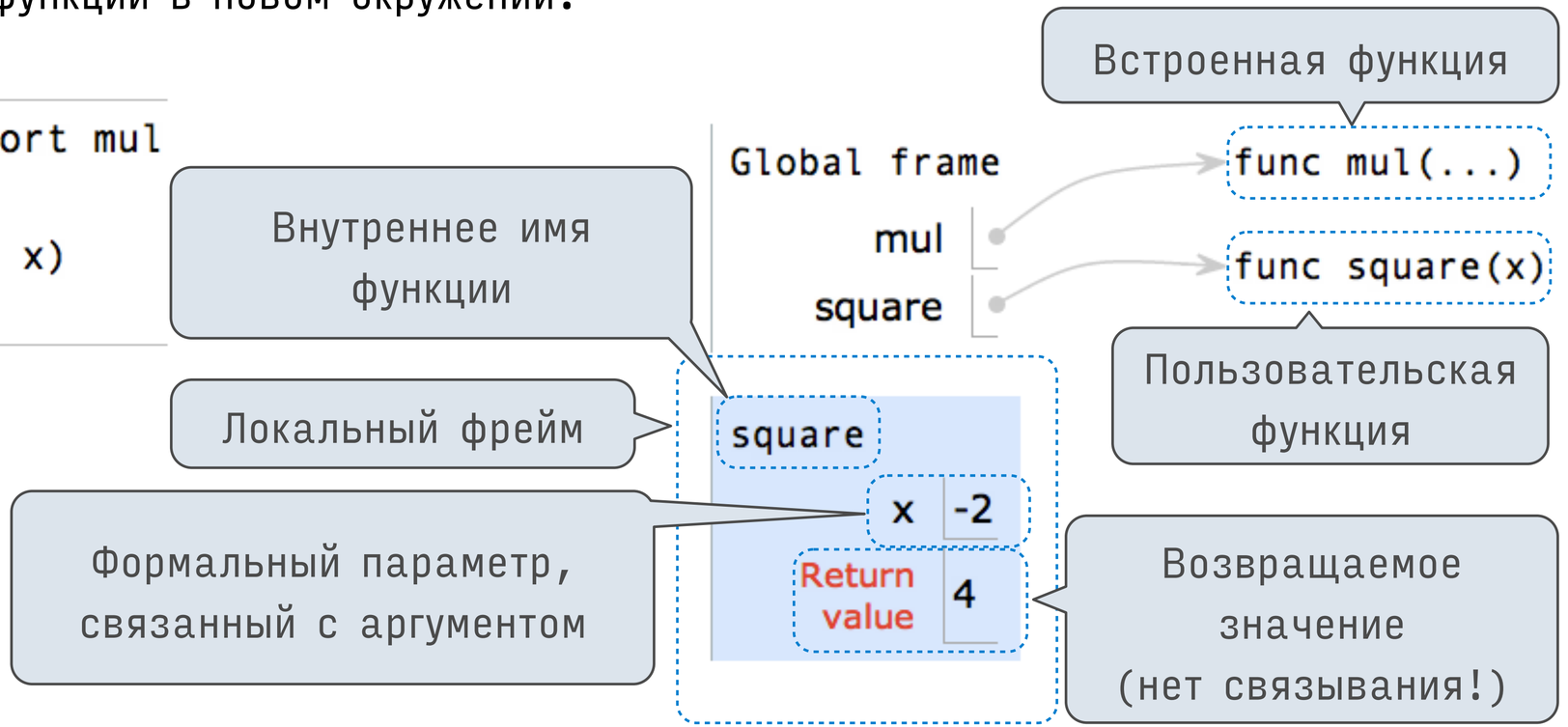
Интерактивная диаграмма

Вызов пользовательских функций

Процедура вызова/применения пользовательской функции (версия 1):

1. Создать новый локальный фрейм, сформировав новое окружение.
2. Связать в этом фрейме формальные параметры функции с аргументами.
3. Выполнить тело функции в новом окружении.

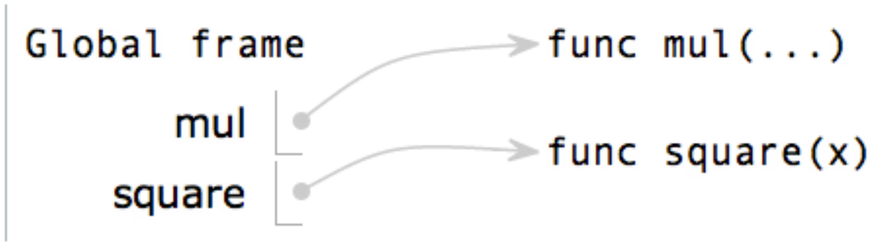
```
1  from operator import mul
2  def square(x):
3      return mul(x, x)
4  square(-2)
```

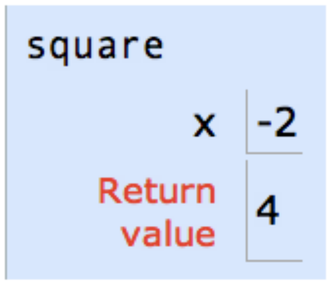


Интерактивная диаграмма

Вызов пользовательских функций

```
1  from operator import mul
2  def square(x):
3      return mul(x, x)
4  square(-2)
```

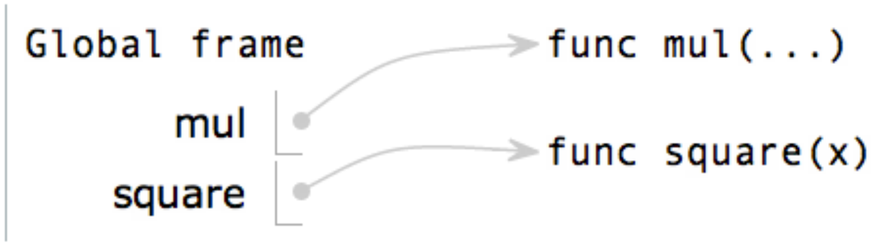




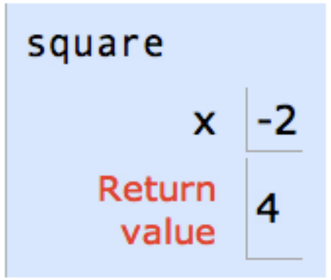
Интерактивная диаграмма

Вызов пользовательских функций

```
1  from operator import mul
2  def square(x):
3      return mul(x, x)
4  square(-2)
```



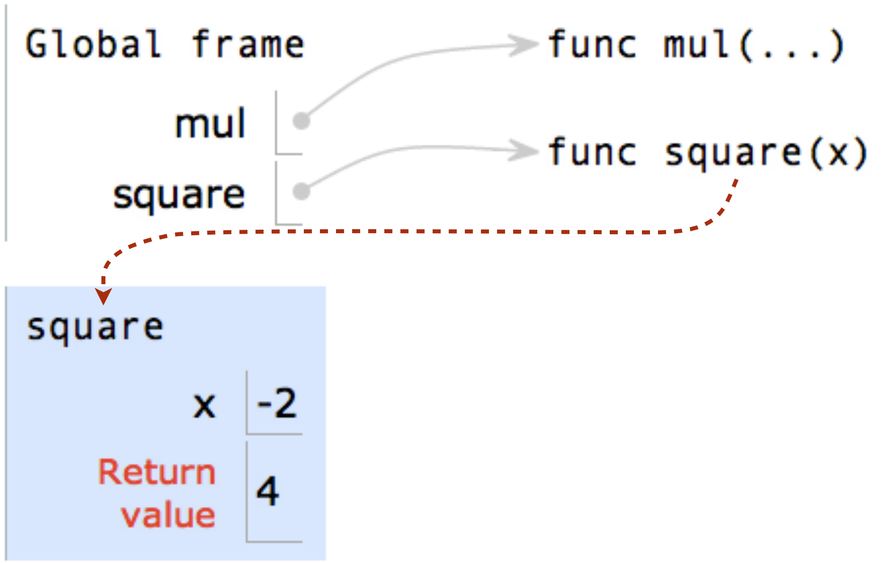
Сигнатура функции содержит всю необходимую информацию для создания фрейма



Интерактивная диаграмма

Вызов пользовательских функций

```
1  from operator import mul
2  def square(x):
3      return mul(x, x)
4  square(-2)
```



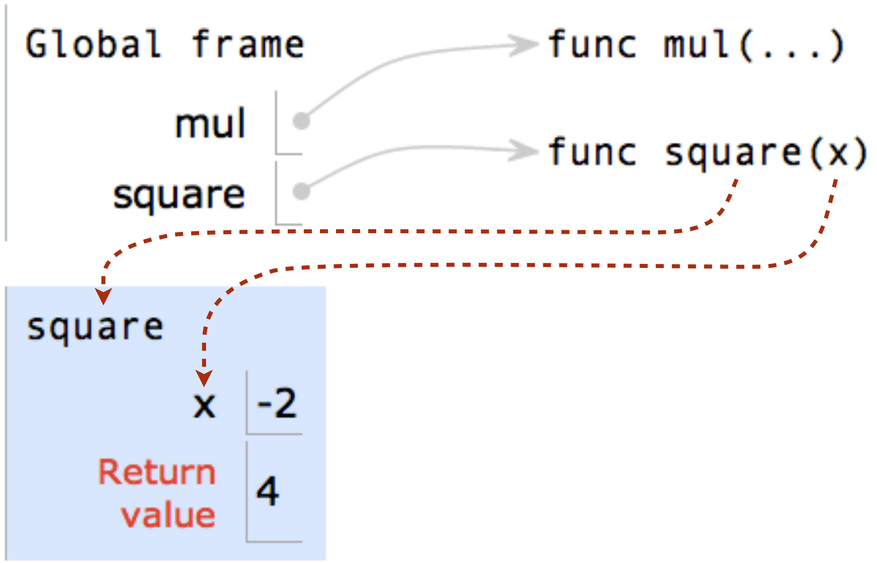
Сигнатура функции содержит всю необходимую информацию для создания фрейма

Интерактивная диаграмма

Вызов пользовательских функций

```
1  from operator import mul
2  def square(x):
3      return mul(x, x)
4  square(-2)
```

Сигнатура функции содержит всю необходимую информацию для создания фрейма



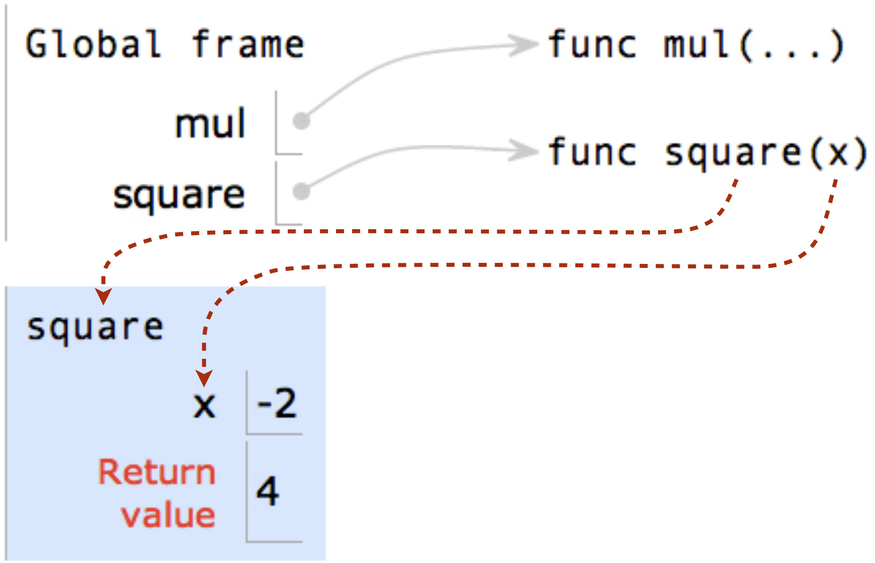
Интерактивная диаграмма

Вызов пользовательских функций

Процедура вызова/применения пользовательской функции (версия 1):

```
1  from operator import mul
2  def square(x):
3      return mul(x, x)
4  square(-2)
```

Сигнатура функции содержит всю необходимую информацию для создания фрейма



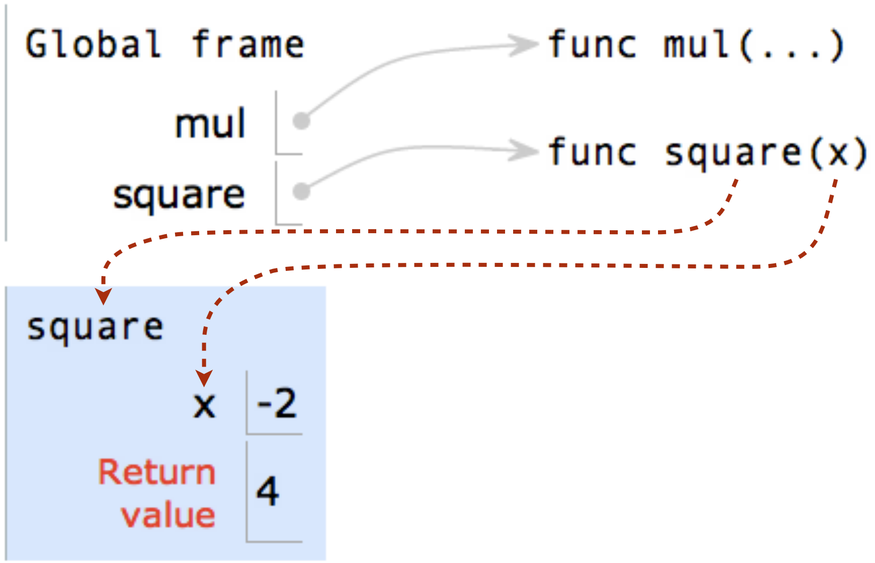
Интерактивная диаграмма

Вызов пользовательских функций

Процедура вызова/применения пользовательской функции (версия 1):

1. Создать новый локальный фрейм, сформировав новое окружение.

```
1  from operator import mul
2  def square(x):
3      return mul(x, x)
4  square(-2)
```



Сигнатура функции содержит всю необходимую информацию для создания фрейма

Интерактивная диаграмма

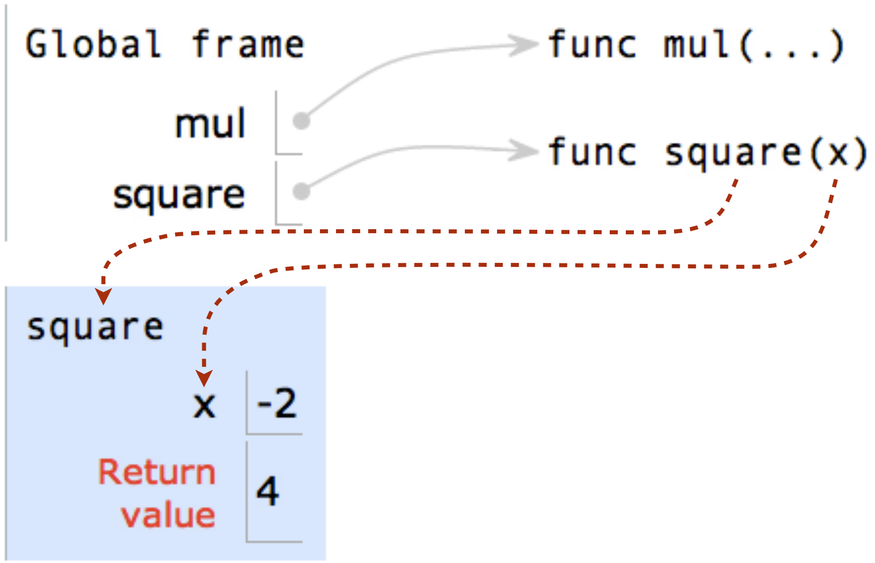
Вызов пользовательских функций

Процедура вызова/применения пользовательской функции (версия 1):

1. Создать новый локальный фрейм, сформировав новое окружение.
2. Связать в этом фрейме формальные параметры функции с аргументами.

```
1  from operator import mul
2  def square(x):
3      return mul(x, x)
4  square(-2)
```

Сигнатура функции содержит всю необходимую информацию для создания фрейма



Интерактивная диаграмма

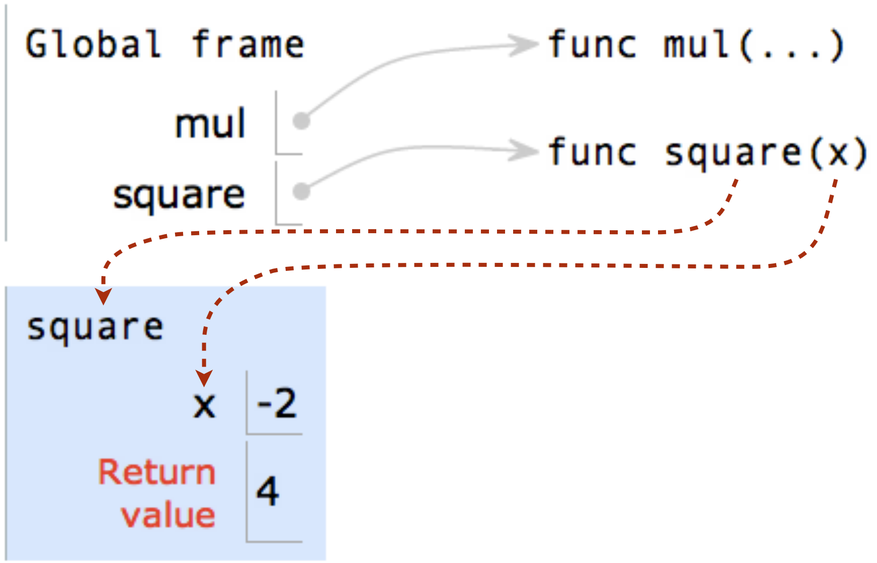
Вызов пользовательских функций

Процедура вызова/применения пользовательской функции (версия 1):

1. Создать новый локальный фрейм, сформировав новое окружение.
2. Связать в этом фрейме формальные параметры функции с аргументами.
3. Выполнить тело функции в новом окружении.

```
1  from operator import mul
2  def square(x):
3      return mul(x, x)
4  square(-2)
```

Сигнатура функции содержит всю необходимую информацию для создания фрейма



Интерактивная диаграмма

Поиск имен в окружении

Поиск имен в окружении

Всякое выражение вычисляется в контексте окружения.

Поиск имен в окружении

Всякое выражение вычисляется в контексте окружения.

Пока что, текущее окружение может состоять из:

Поиск имен в окружении

Всякое выражение вычисляется в контексте окружения.

Пока что, текущее окружение может состоять из:

- Глобального фрейма, или

Поиск имен в окружении

Всякое выражение вычисляется в контексте окружения.

Пока что, текущее окружение может состоять из:

- Глобального фрейма, или
- Локального фрейма и, далее, глобального фрейма.

Поиск имен в окружении

Всякое выражение вычисляется в контексте окружения.

Пока что, текущее окружение может состоять из:

- Глобального фрейма, или
- Локального фрейма и, далее, глобального фрейма.

Две самые важные мысли в этой лекции:

Поиск имен в окружении

Всякое выражение вычисляется в контексте окружения.

Пока что, текущее окружение может состоять из:

- Глобального фрейма, или
- Локального фрейма и, далее, глобального фрейма.

Две самые важные мысли в этой лекции:

- Окружение — это последовательность фреймов.

Поиск имен в окружении

Всякое выражение вычисляется в контексте окружения.

Пока что, текущее окружение может состоять из:

- Глобального фрейма, или
- Локального фрейма и, далее, глобального фрейма.

Две самые важные мысли в этой лекции:

- Окружение — это последовательность фреймов.
- Значение имени в текущем окружении берётся из первого фрейма, в котором это имя встречается.

Поиск имен в окружении

Всякое выражение вычисляется в контексте окружения.

Пока что, текущее окружение может состоять из:

- Глобального фрейма, или
- Локального фрейма и, далее, глобального фрейма.

Две самые важные мысли в этой лекции:

- Окружение — это последовательность фреймов.
- Значение имени в текущем окружении берётся из первого фрейма, в котором это имя встречается.

То есть для поиска некоторого имени внутри тела функции **square:**

Поиск имен в окружении

Всякое выражение вычисляется в контексте окружения.

Пока что, текущее окружение может состоять из:

- Глобального фрейма, или
- Локального фрейма и, далее, глобального фрейма.

Две самые важные мысли в этой лекции:

- Окружение — это последовательность фреймов.
- Значение имени в текущем окружении берётся из первого фрейма, в котором это имя встречается.

То есть для поиска некоторого имени внутри тела функции **square:**

- ищем это имя в локальном фрейме;

Поиск имен в окружении

Всякое выражение вычисляется в контексте окружения.

Пока что, текущее окружение может состоять из:

- Глобального фрейма, или
- Локального фрейма и, далее, глобального фрейма.

Две самые важные мысли в этой лекции:

- Окружение — это последовательность фреймов.
- Значение имени в текущем окружении берётся из первого фрейма, в котором это имя встречается.

То есть для поиска некоторого имени внутри тела функции **square:**

- ищем это имя в локальном фрейме;
- если не находим, то ищем это имя в глобальном фрейме
  (встроенные имена вроде «max» также находятся в глобальном фрейме,
  однако они не отображаются на диаграмме окружения).

Поиск имен в окружении

Всякое выражение вычисляется в контексте окружения.

Пока что, текущее окружение может состоять из:

- Глобального фрейма, или
- Локального фрейма и, далее, глобального фрейма.

Две самые важные мысли в этой лекции:

- Окружение — это последовательность фреймов.
- Значение имени в текущем окружении берётся из первого фрейма, в котором это имя встречается.

То есть для поиска некоторого имени внутри тела функции **square:**

- ищем это имя в локальном фрейме;
- если не находим, то ищем это имя в глобальном фрейме
  (встроенные имена вроде «max» также находятся в глобальном фрейме,
  однако они не отображаются на диаграмме окружения).

(Пример)

Print и None

(Пример)

None указывает, что ничего не возвращается

None указывает, что ничего не возвращается

Специальное значение **None** означает в Python «ничто».

None указывает, что ничего не возвращается

Специальное значение **None** означает в Python «ничто».

Функция без **return** неявно возвращает **None**.

None указывает, что ничего не возвращается

Специальное значение **None** означает в Python «ничто».

Функция без **return** неявно возвращает **None**.

Внимание: **None** *не отображается* интерпретатором в качестве значения выражения.

None указывает, что ничего не возвращается

Специальное значение **None** означает в Python «ничто».

Функция без **return** неявно возвращает **None**.

Внимание: **None** *не отображается* интерпретатором в качестве значения выражения.

```
>>> def does_not_square(x):
...     x * x
...
```

None указывает, что ничего не возвращается

Специальное значение **None** означает в Python «ничто».

Функция без **return** неявно возвращает **None**.

Внимание: **None** *не отображается* интерпретатором в качестве значения выражения.

```
>>> def does_not_square(x):
...     x * x
...
```

None указывает, что ничего не возвращается

Специальное значение **None** означает в Python «ничто».

Функция без **return** неявно возвращает **None**.

Внимание: **None** *не отображается* интерпретатором в качестве значения выражения.

```
>>> def does_not_square(x):
...     x * x
...
>>> does_not_square(4)
```

None указывает, что ничего не возвращается

Специальное значение **None** означает в Python «ничто».

Функция без **return** неявно возвращает **None**.

Внимание: **None** *не отображается* интерпретатором в качестве значения выражения.

```
>>> def does_not_square(x):
...     x * x
...
>>> does_not_square(4)
```

Нет **return**

Значение **None** не показывается

None указывает, что ничего не возвращается

Специальное значение **None** означает в Python «ничто».

Функция без **return** неявно возвращает **None**.

Внимание: **None** *не отображается* интерпретатором в качестве значения выражения.

```
>>> def does_not_square(x):
...     x * x
...
>>> does_not_square(4)
>>> sixteen = does_not_square(4)
```

Нет **return**

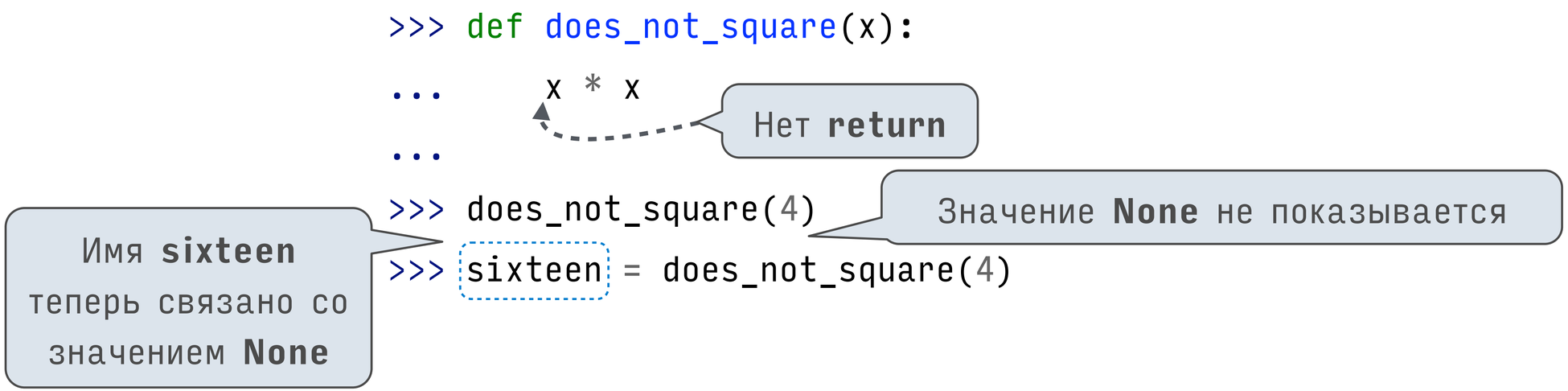
Значение **None** не показывается

None указывает, что ничего не возвращается

Специальное значение **None** означает в Python «ничто».

Функция без **return** неявно возвращает **None**.

Внимание: **None** *не отображается* интерпретатором в качестве значения выражения.



None указывает, что ничего не возвращается

Специальное значение **None** означает в Python «ничто».

Функция без **return** неявно возвращает **None**.

Внимание: **None** *не отображается* интерпретатором в качестве значения выражения.

```
>>> def does_not_square(x):
...     x * x
...
>>> does_not_square(4)
>>> sixteen = does_not_square(4)
>>> sixteen + 4
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'NoneType' and 'int'
```

Нет **return**

Значение **None** не показывается

Имя **sixteen** теперь связано со значением **None**

Чистые и нечистые функции

Чистые функции

просто возвращают значения

Нечистые функции

имеют побочные эффекты

Чистые и нечистые функции

Чистые функции

просто возвращают значения

Нечистые функции

имеют побочные эффекты

Чистые и нечистые функции

Чистые функции

просто возвращают значения

Нечистые функции

имеют побочные эффекты

Чистые и нечистые функции

Чистые функции

просто возвращают значения

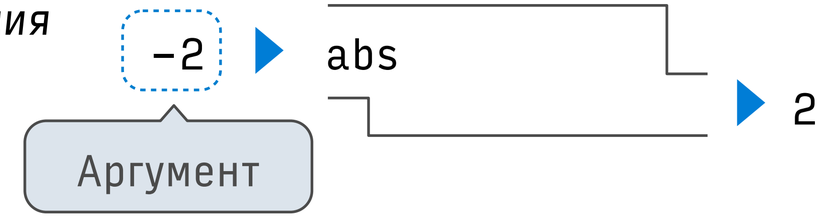
Нечистые функции

имеют побочные эффекты

Чистые и нечистые функции

Чистые функции

просто возвращают значения



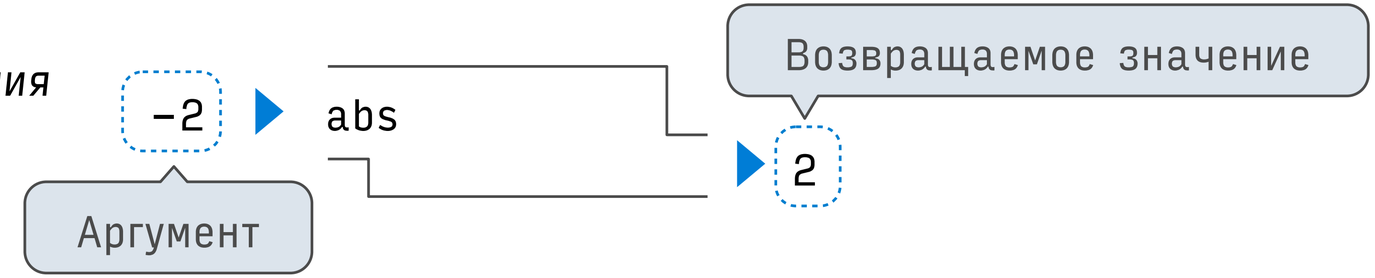
Нечистые функции

имеют побочные эффекты

Чистые и нечистые функции

Чистые функции

просто возвращают значения



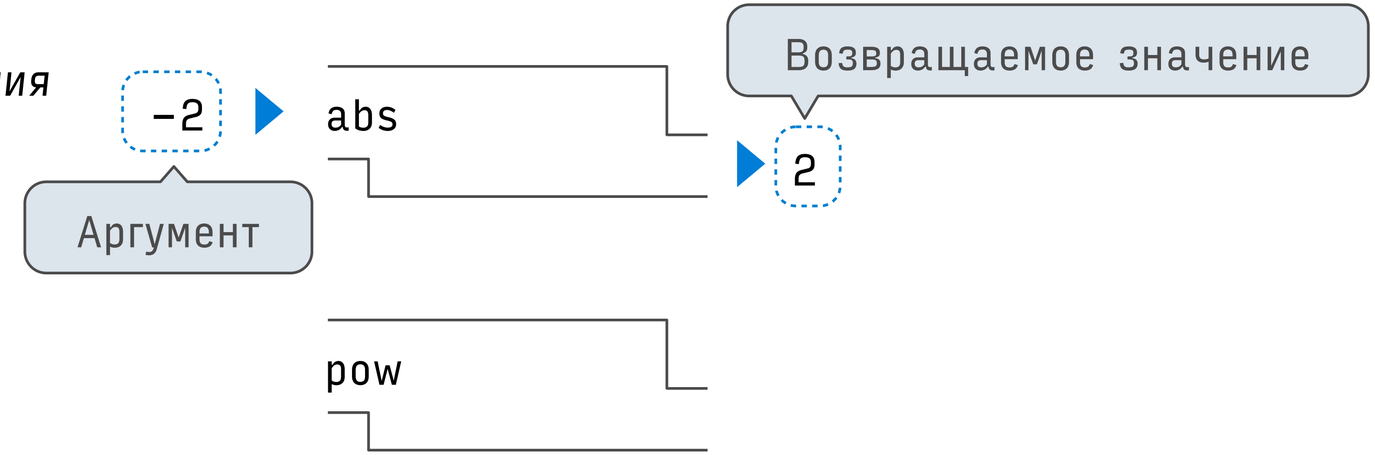
Нечистые функции

имеют побочные эффекты

Чистые и нечистые функции

Чистые функции

просто возвращают значения



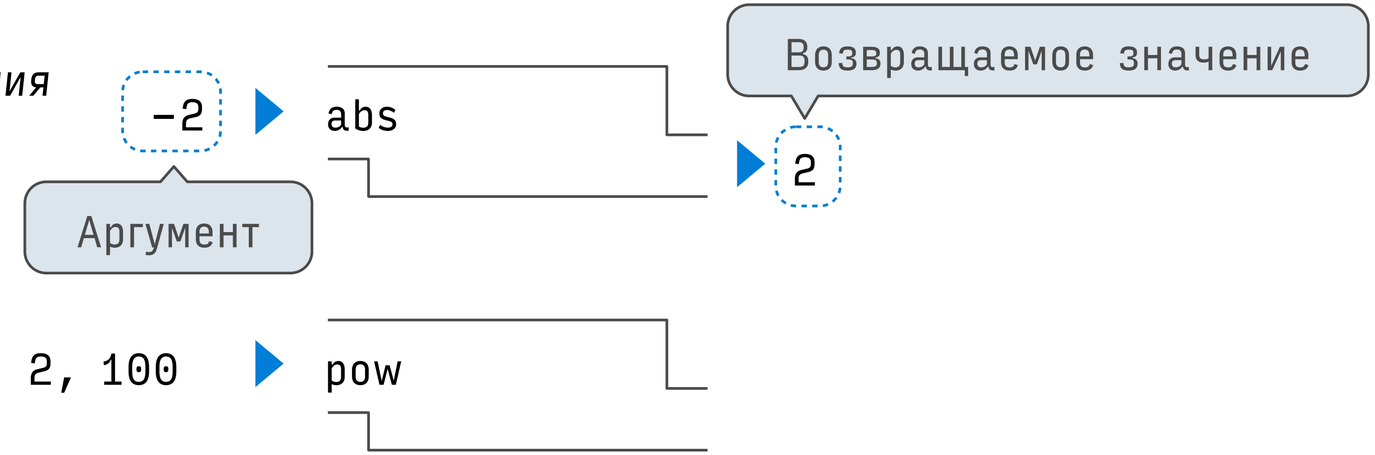
Нечистые функции

имеют побочные эффекты

Чистые и нечистые функции

Чистые функции

просто возвращают значения



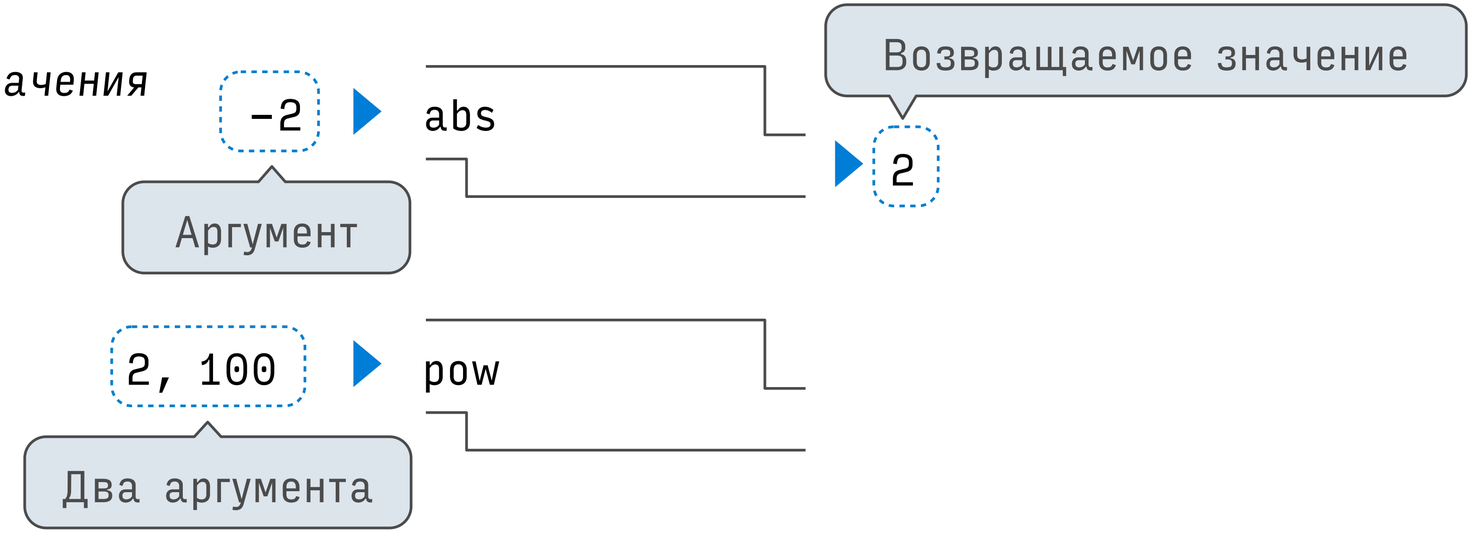
Нечистые функции

имеют побочные эффекты

Чистые и нечистые функции

Чистые функции

просто возвращают значения



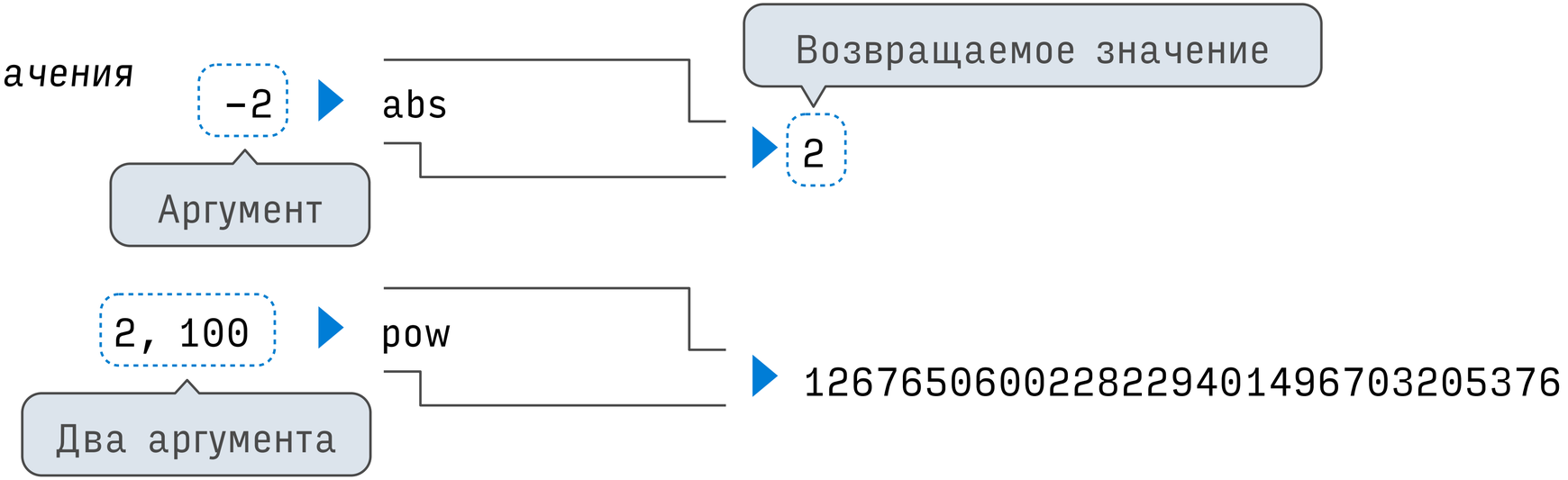
Нечистые функции

имеют побочные эффекты

Чистые и нечистые функции

Чистые функции

просто возвращают значения



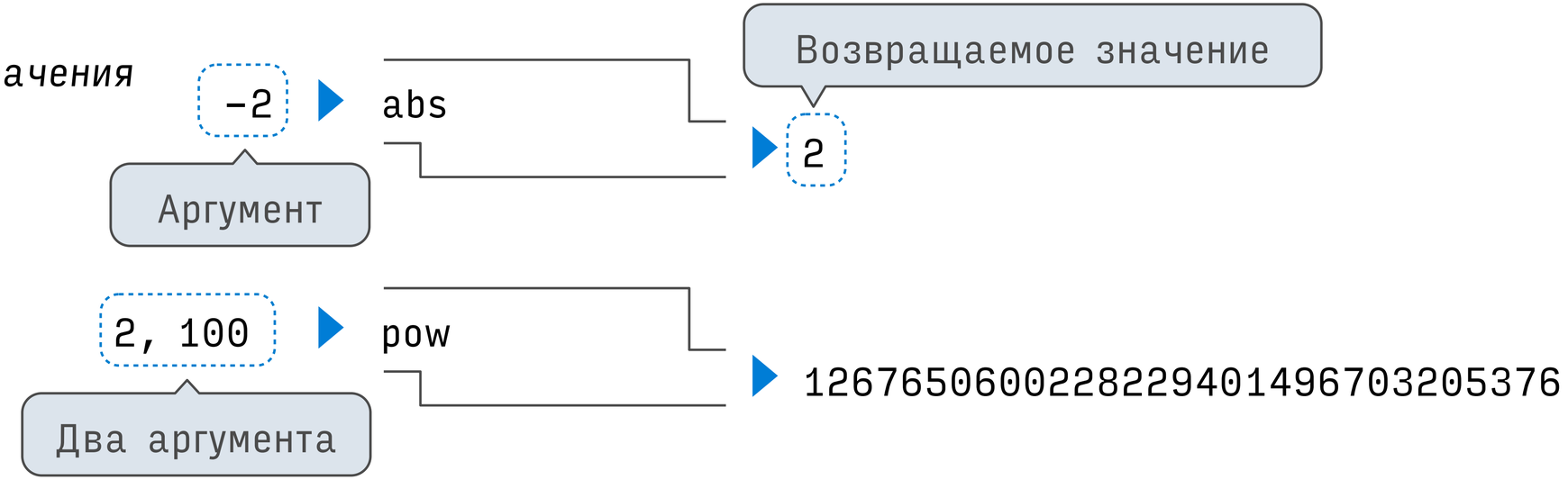
Нечистые функции

имеют побочные эффекты

Чистые и нечистые функции

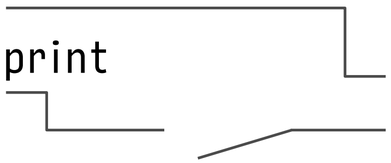
Чистые функции

просто возвращают значения



Нечистые функции

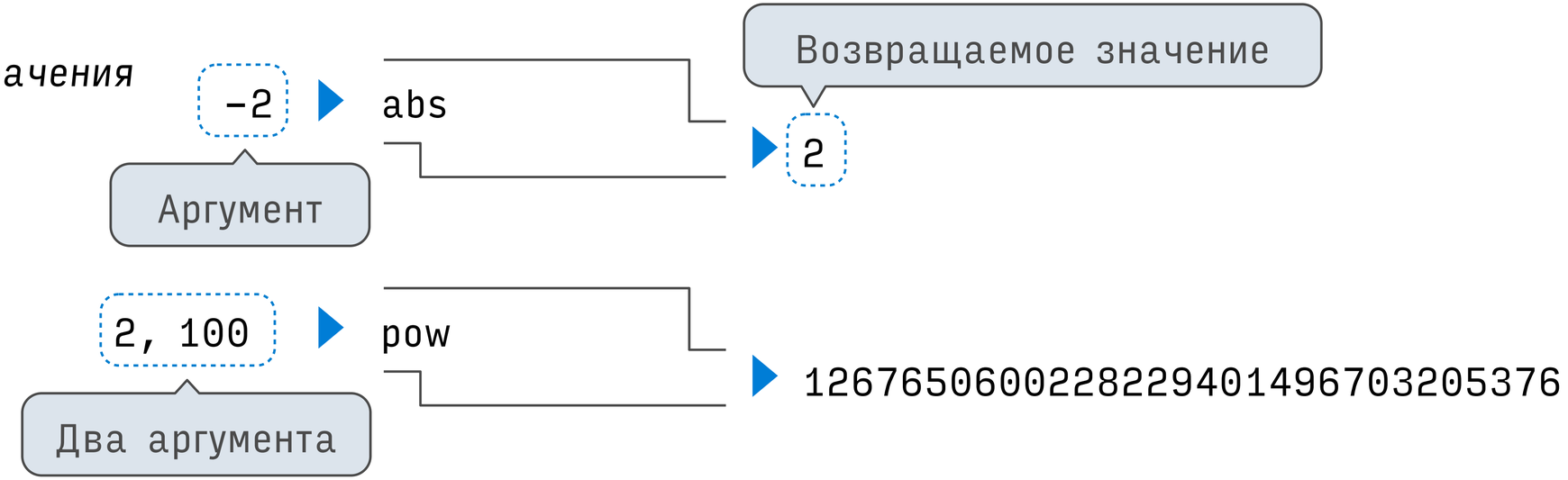
имеют побочные эффекты



Чистые и нечистые функции

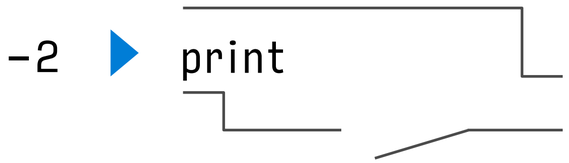
Чистые функции

просто возвращают значения



Нечистые функции

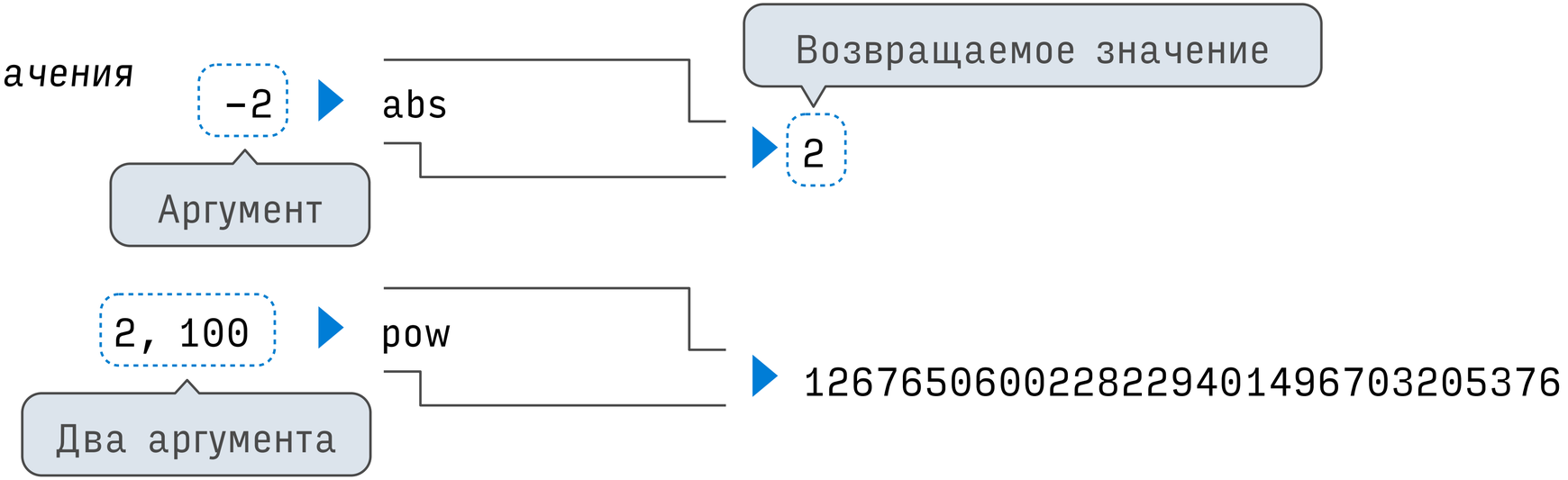
имеют побочные эффекты



Чистые и нечистые функции

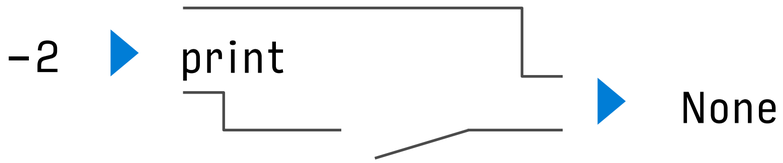
Чистые функции

просто возвращают значения



Нечистые функции

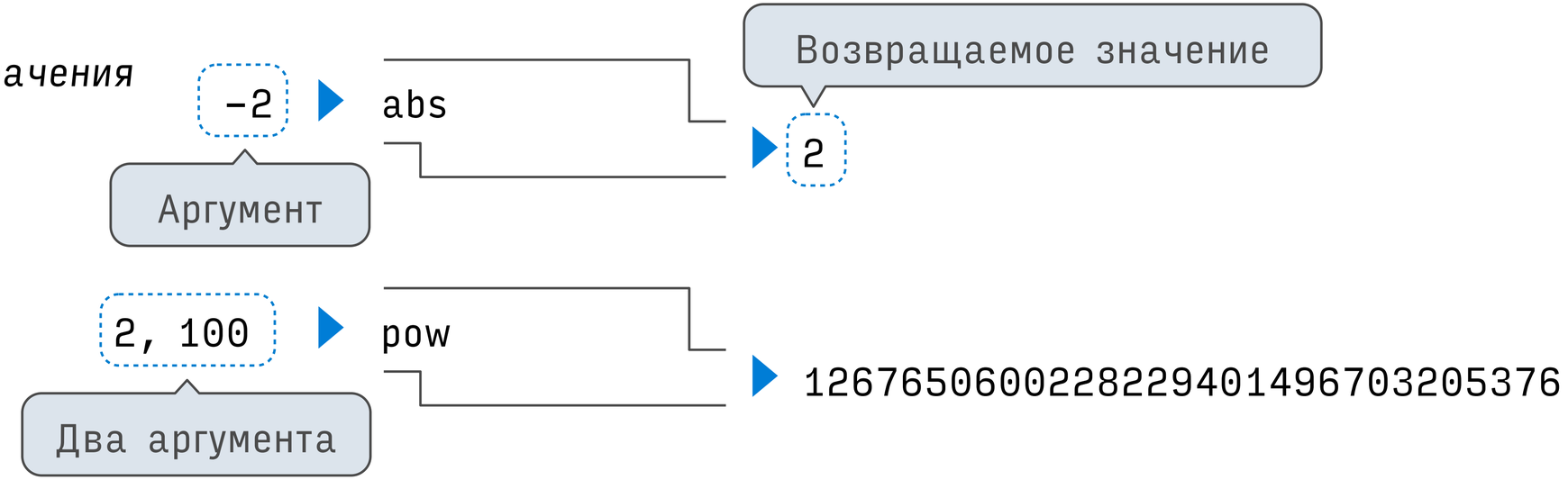
имеют побочные эффекты



Чистые и нечистые функции

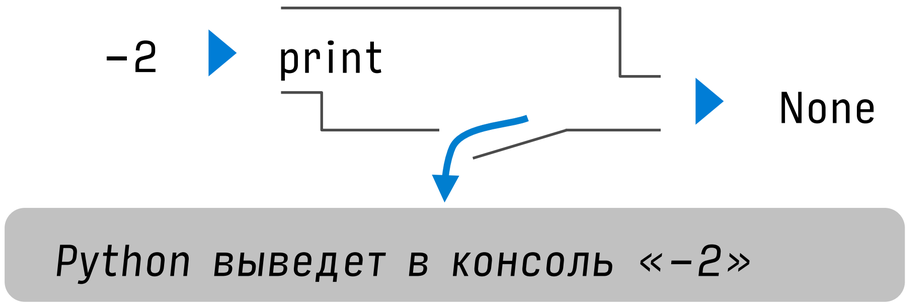
Чистые функции

просто возвращают значения



Нечистые функции

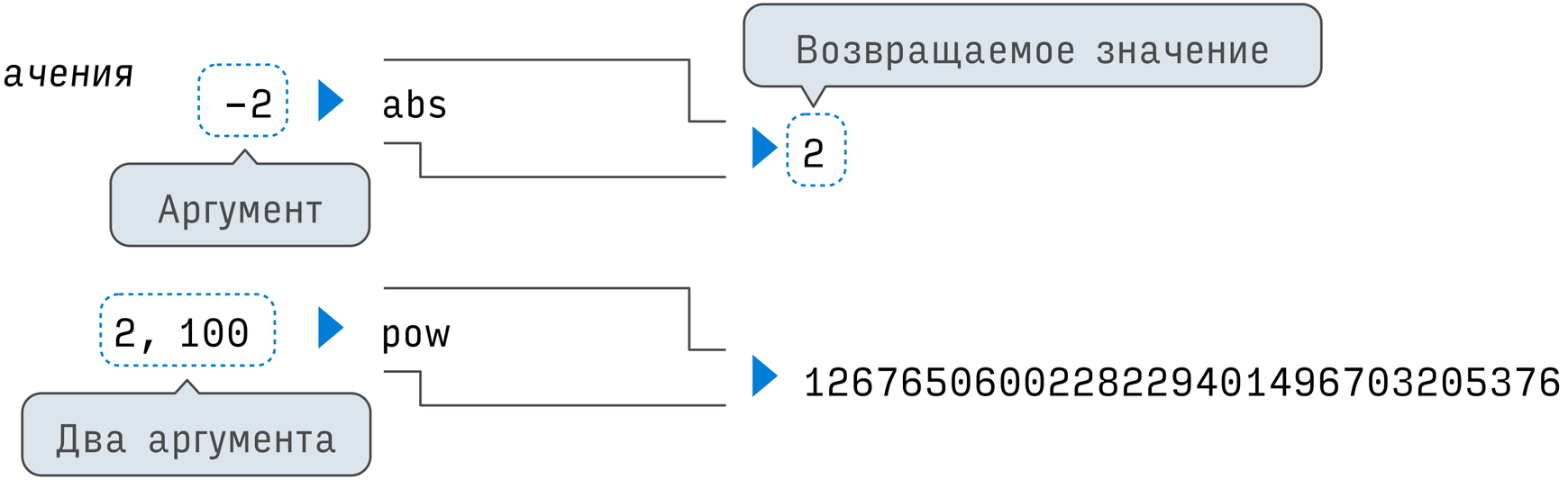
имеют побочные эффекты



Чистые и нечистые функции

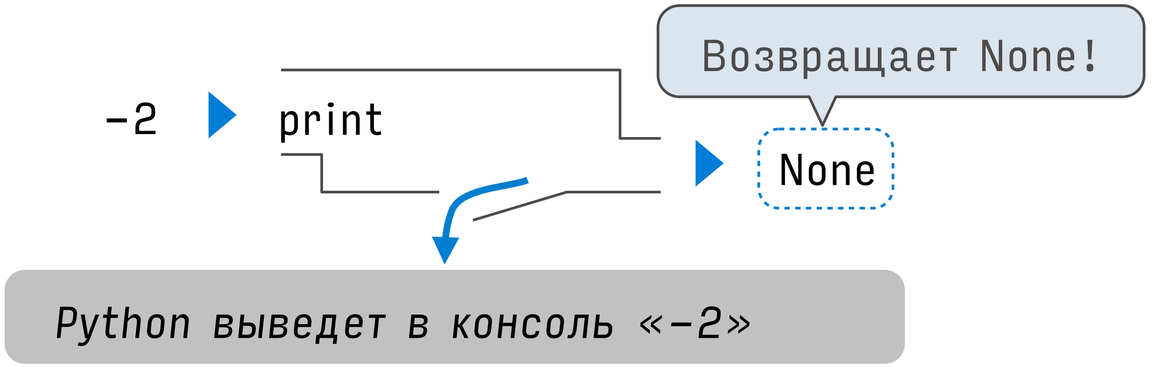
Чистые функции

просто возвращают значения



Нечистые функции

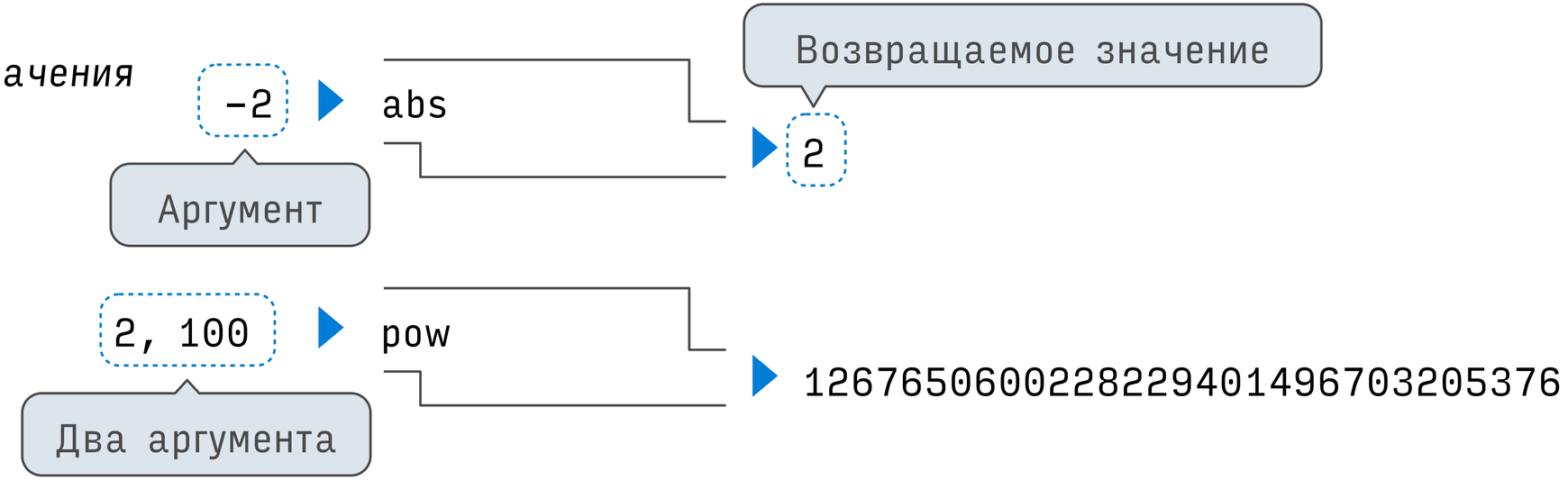
имеют побочные эффекты



Чистые и нечистые функции

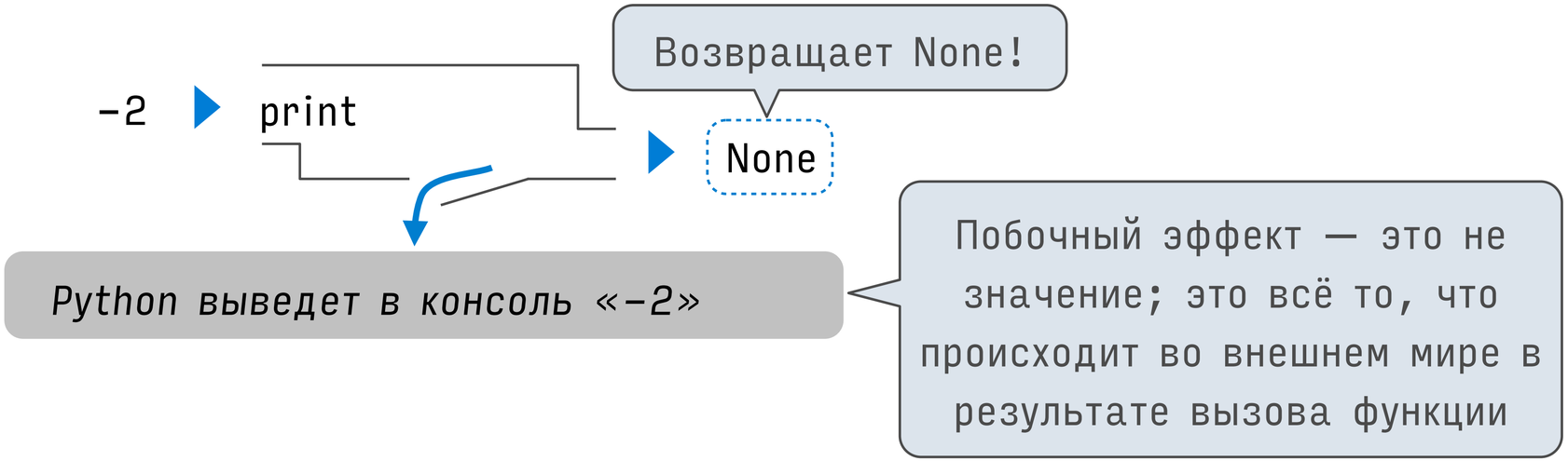
Чистые функции

просто возвращают значения



Нечистые функции

имеют побочные эффекты

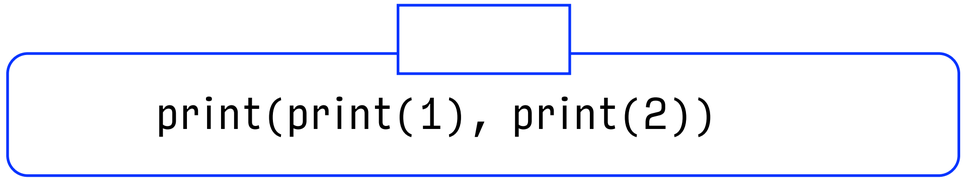


Вложенные выражения с **print**

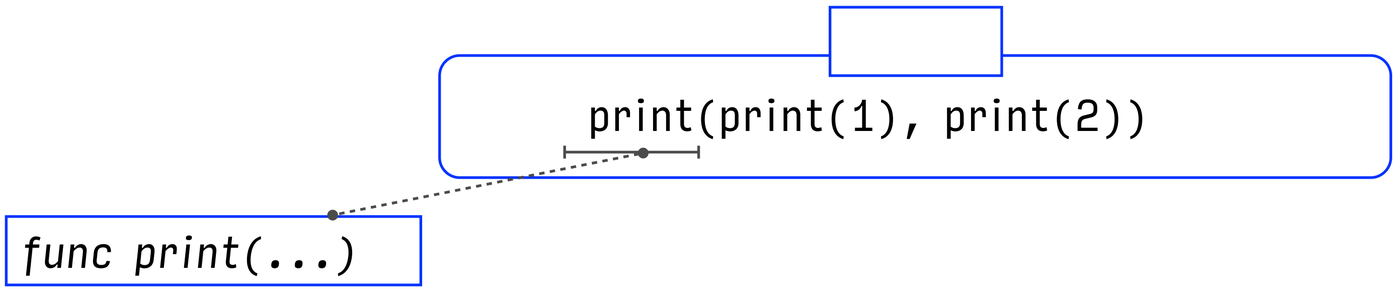
Вложенные выражения с **print**

```
print(print(1), print(2))
```

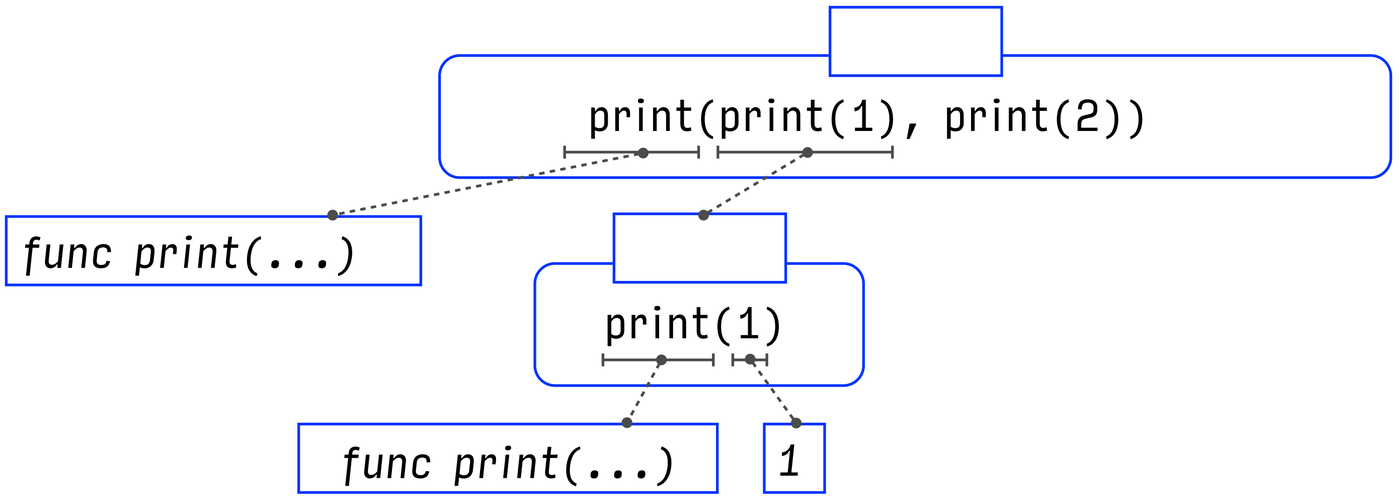
Вложенные выражения с **print**



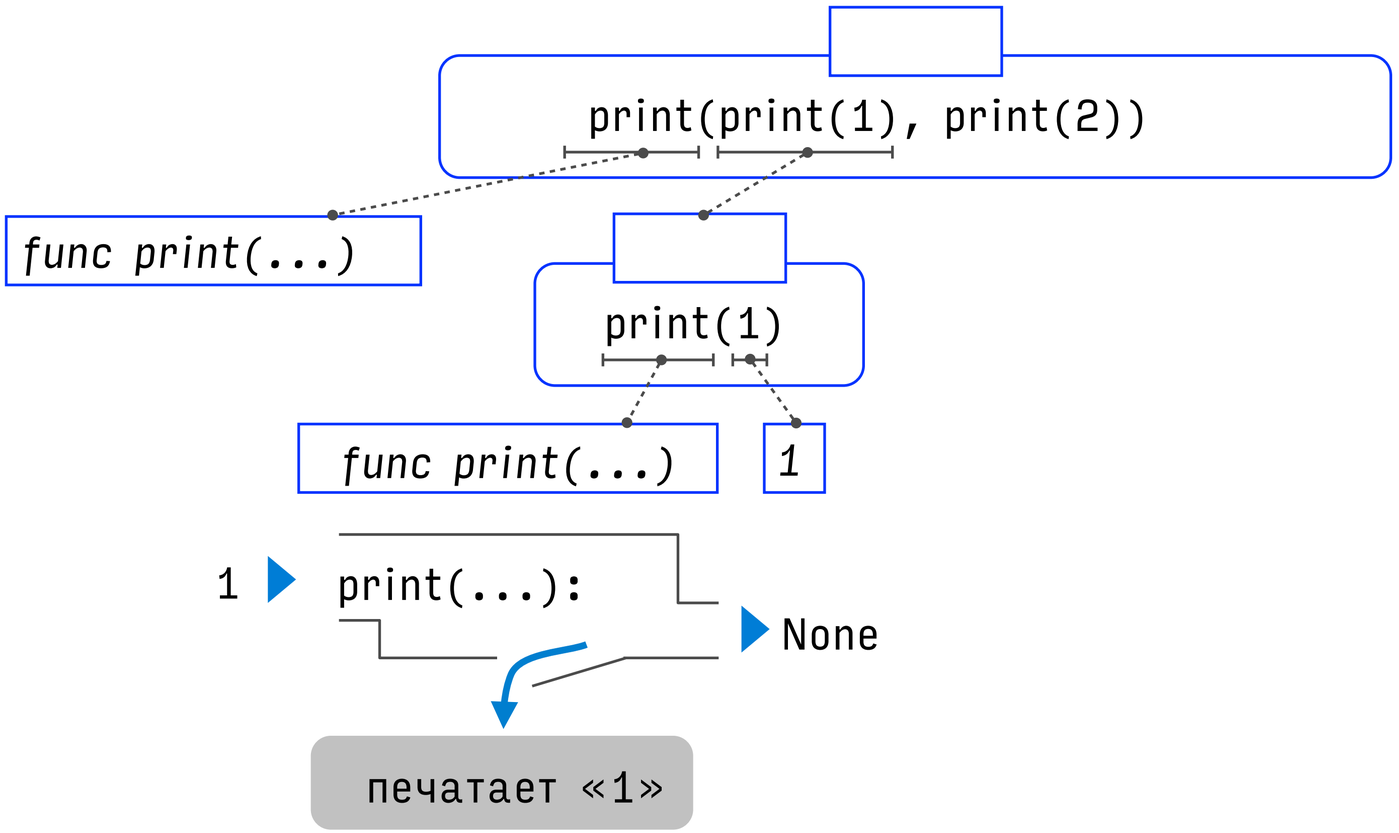
Вложенные выражения с **print**



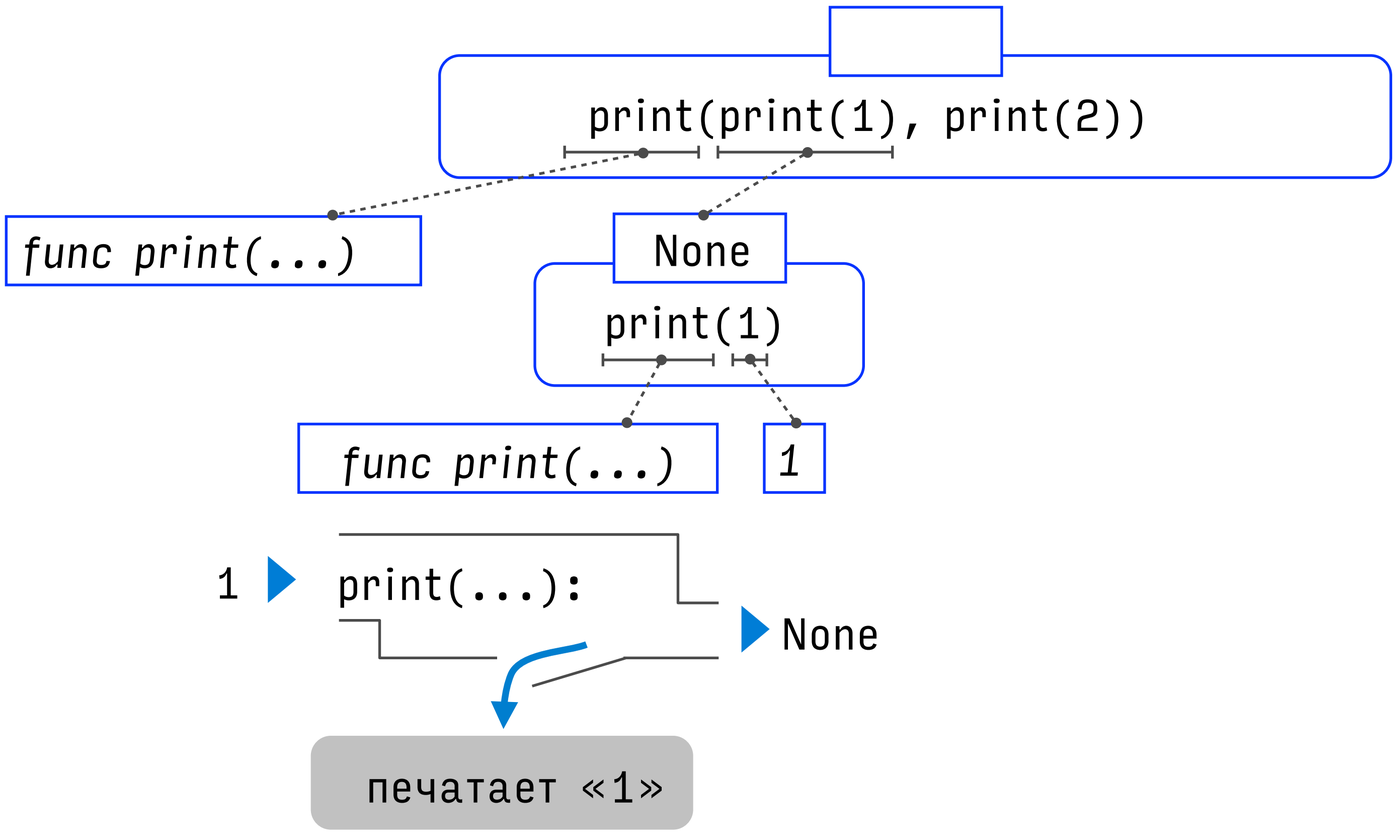
Вложенные выражения с **print**



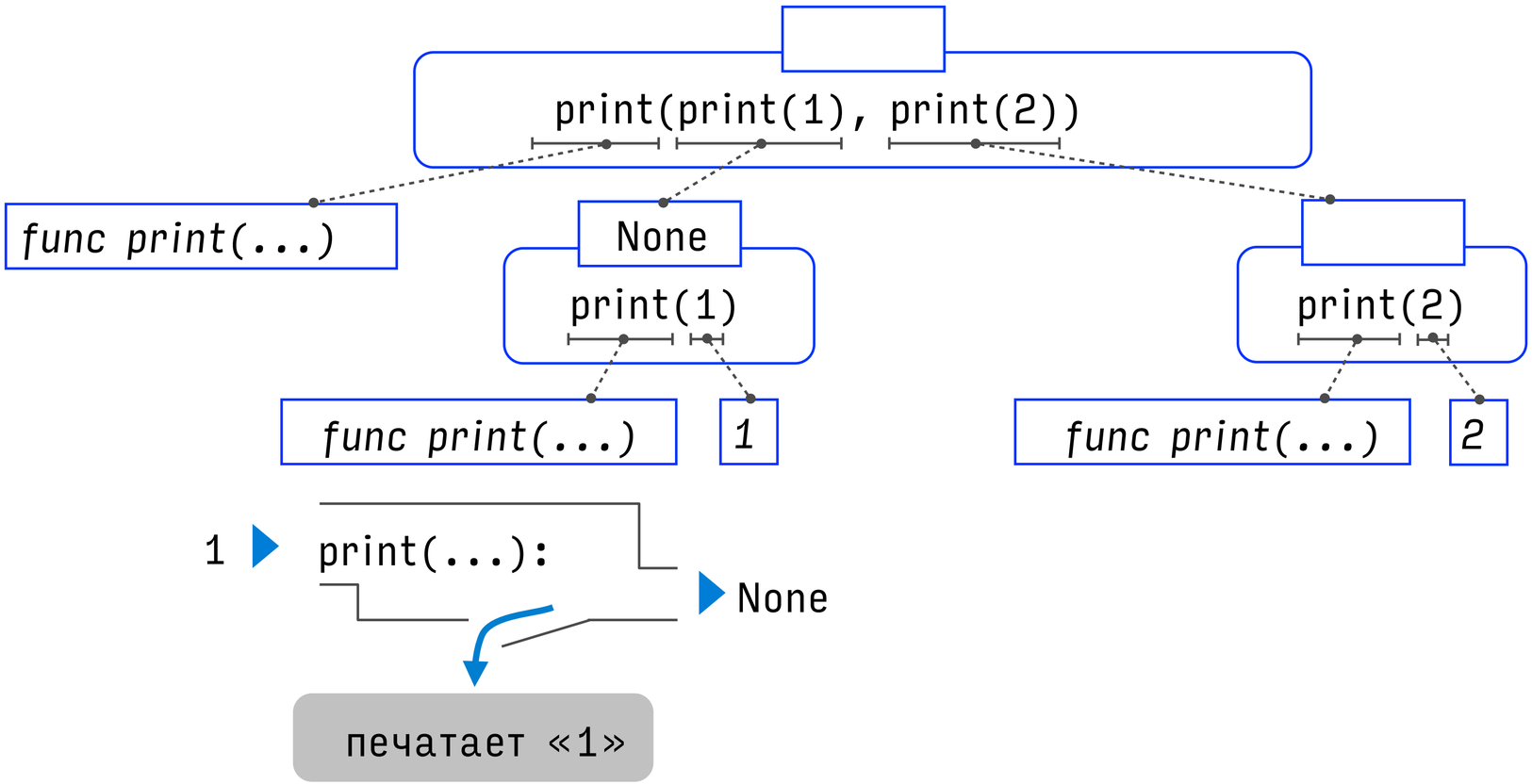
Вложенные выражения с **print**



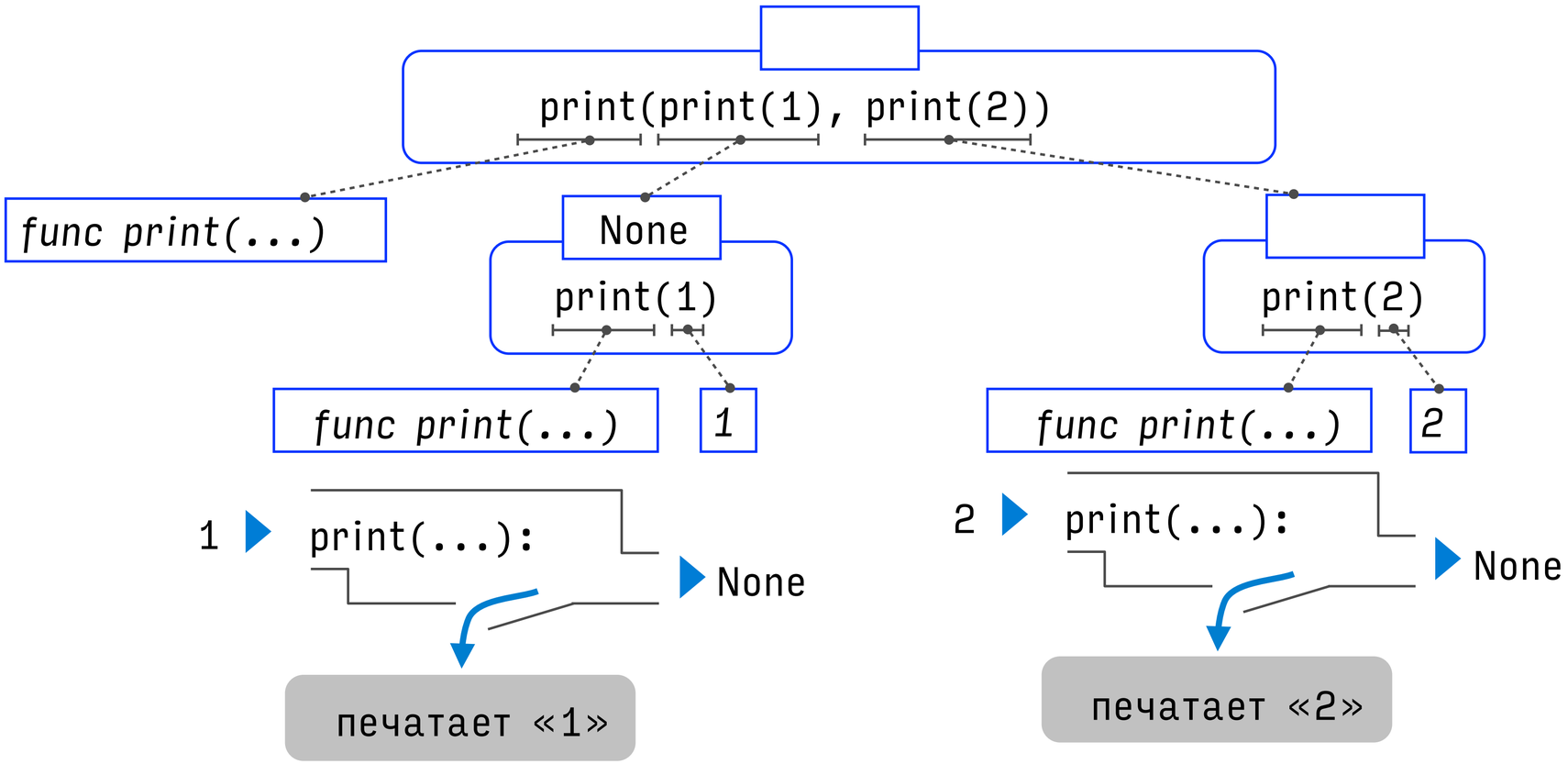
Вложенные выражения с **print**



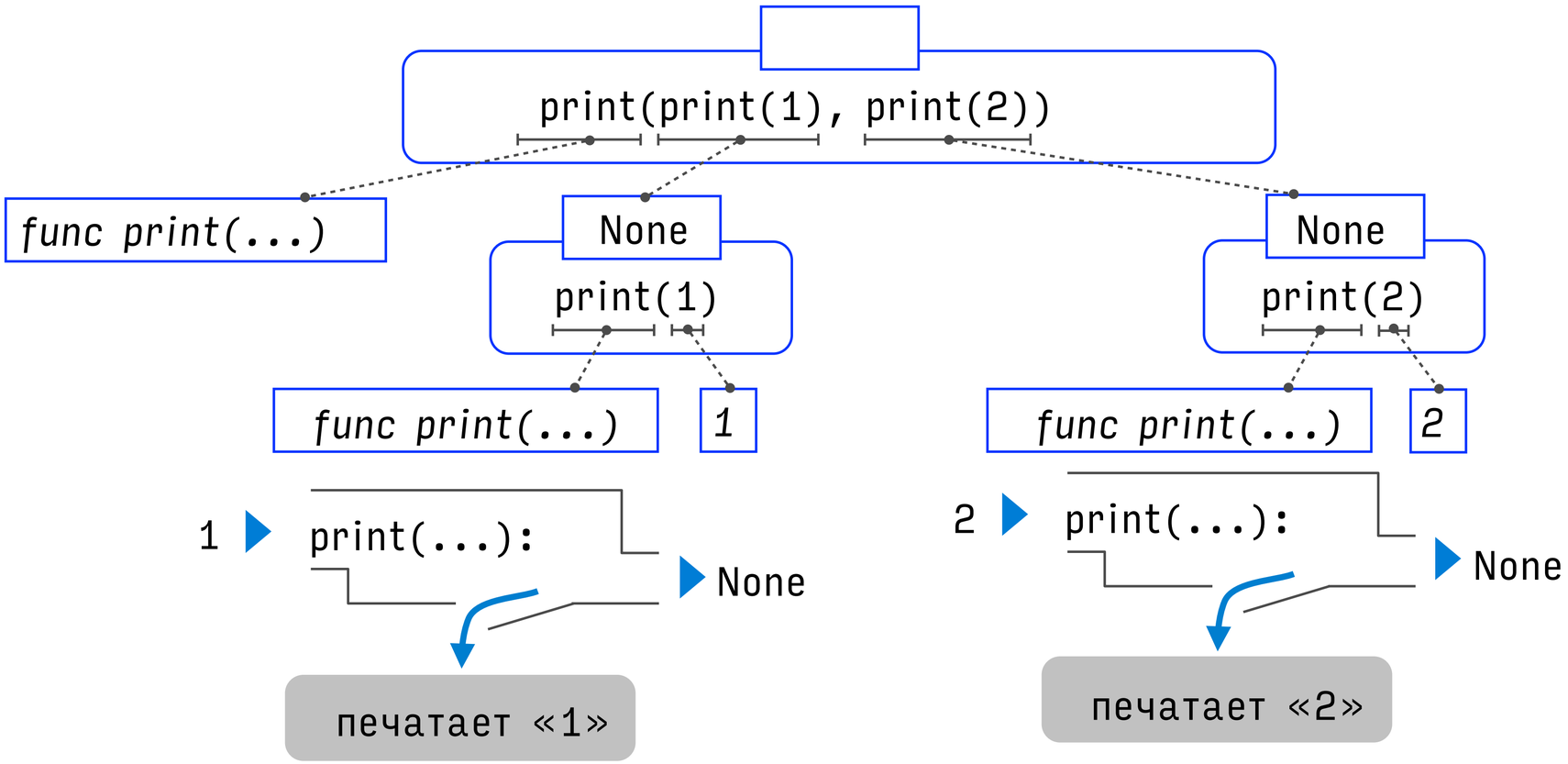
Вложенные выражения с **print**



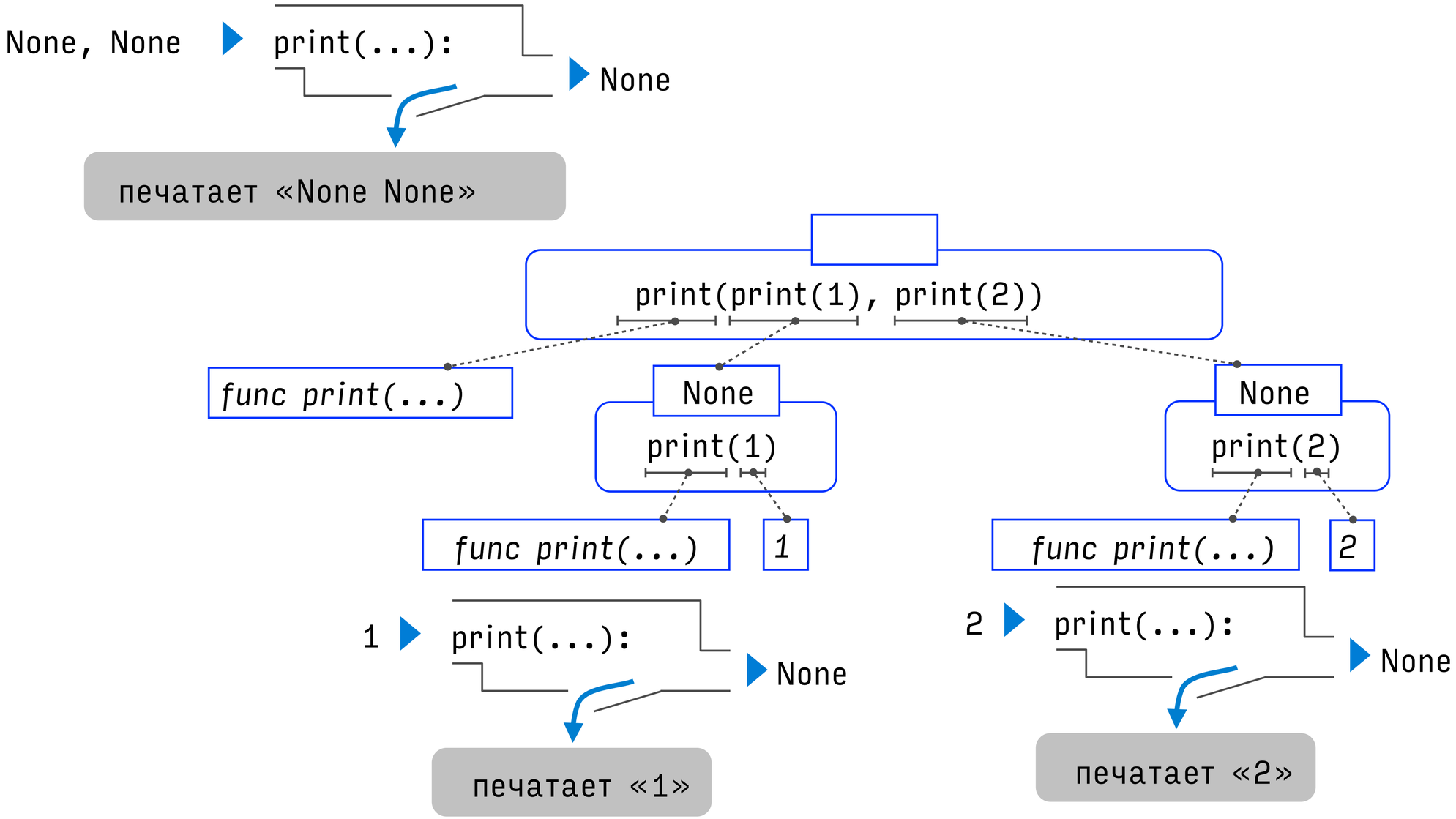
Вложенные выражения с **print**



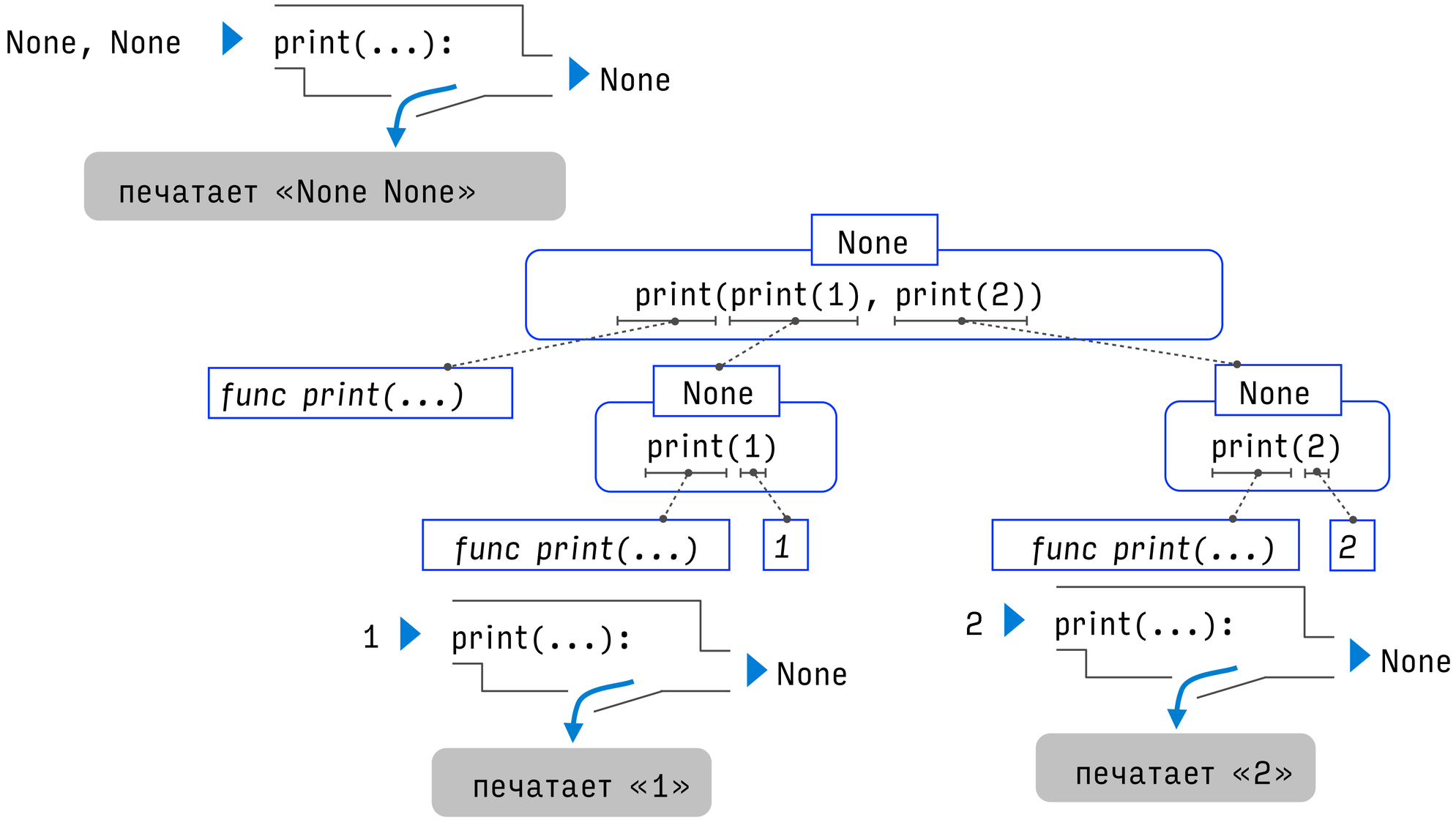
Вложенные выражения с **print**



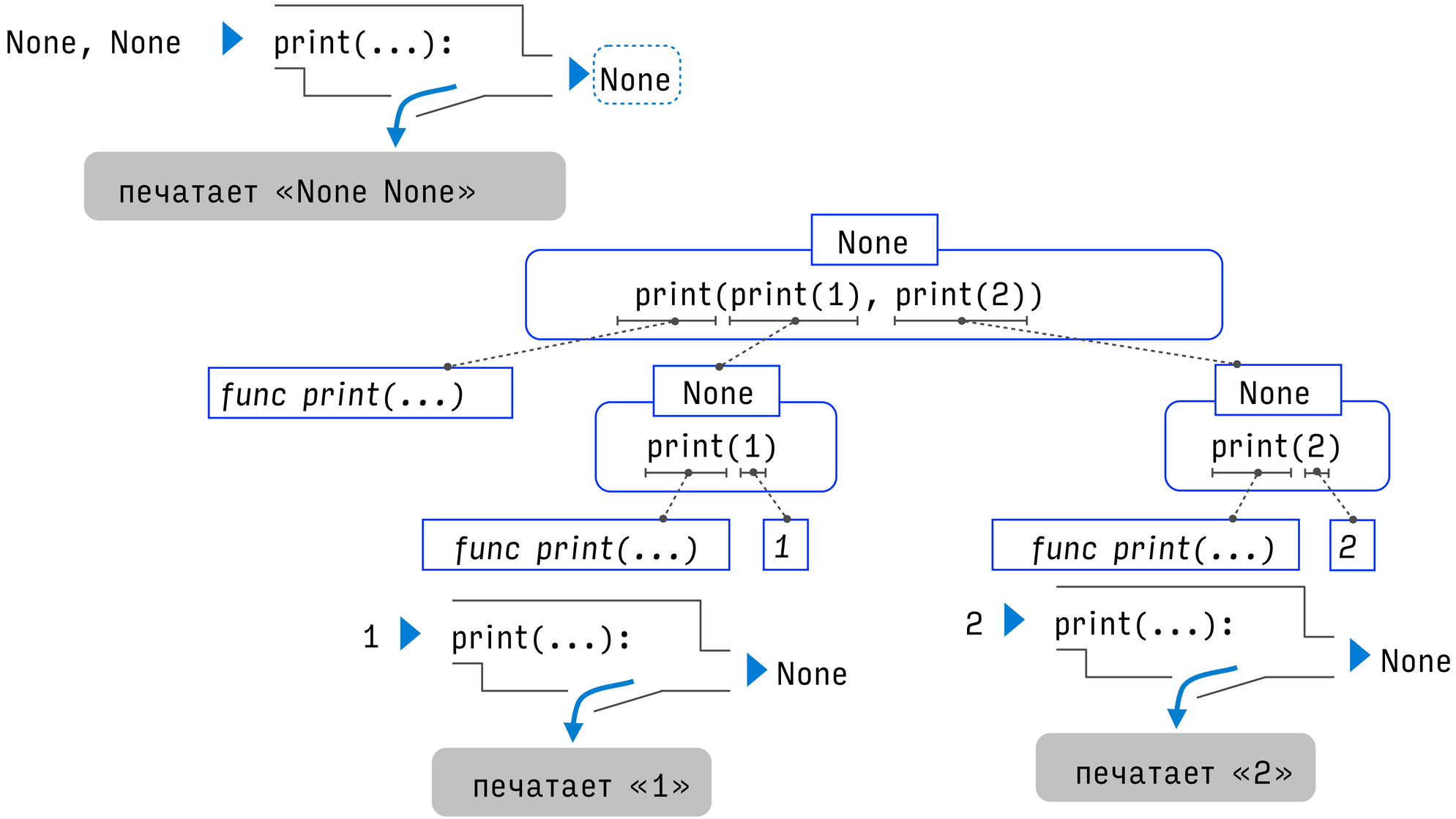
Вложенные выражения с **print**



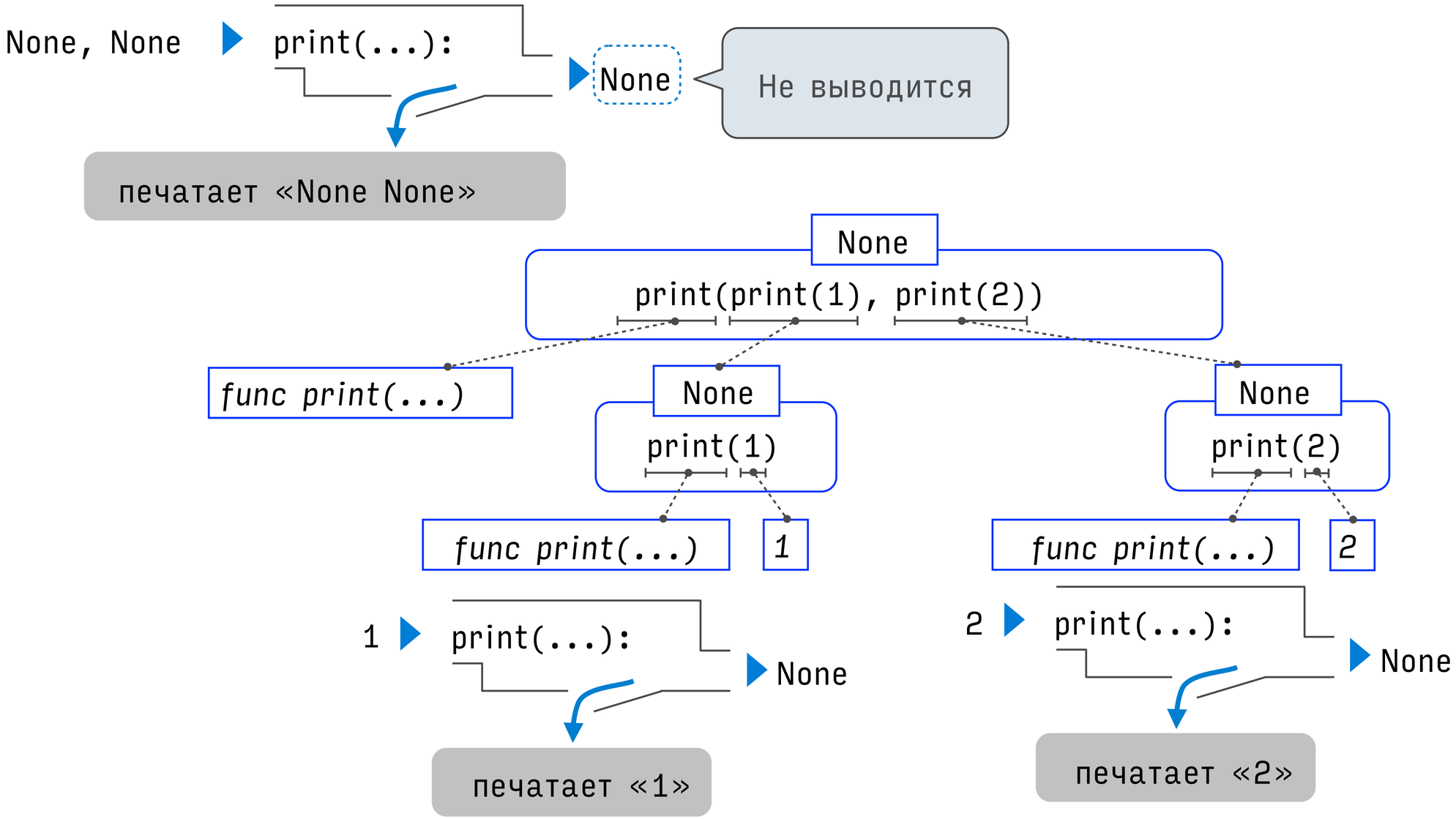
Вложенные выражения с **print**



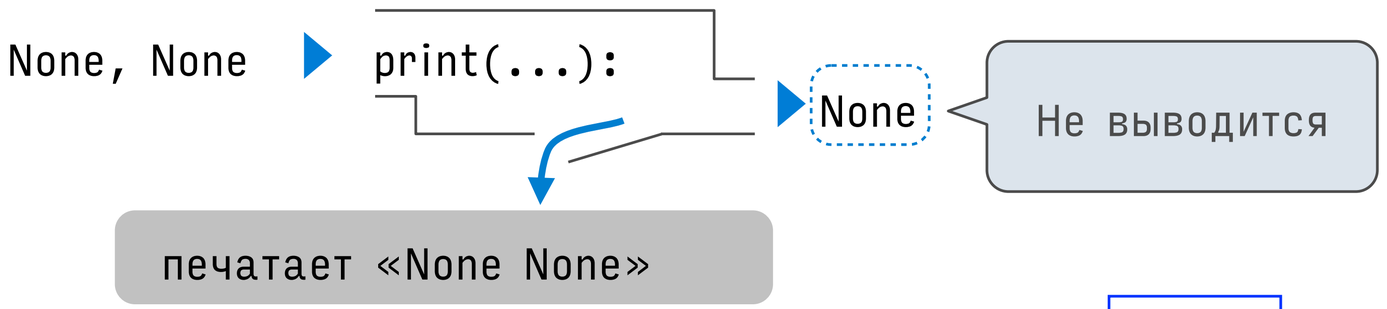
Вложенные выражения с **print**



Вложенные выражения с **print**



Вложенные выражения с **print**




```
>>> print(print(1), print(2))
1
2
None None
```