

Лекция 8

Абстракция

Функциональные абстракции

```
def square(x):  
    return mul(x, x)
```

```
def sum_squares(x, y):  
    return square(x) + square(y)
```

Что «sum_squares» должна знать про «square»?

- «square» принимает один аргумент? **Да**
- «square» имеет внутреннее имя «square»? **Нет**
- «square» вычисляет квадрат аргумента? **Да**
- «square» вычисляет квадрат, используя «mul»? **Нет**

```
def square(x):  
    return pow(x, 2)
```

```
def square(x):  
    return mul(x, x-1) + x
```

Независимо от реализации функции «square», функция «sum_squares» продолжит работать так же хорошо.

Выбор имен

Обычно имена не влияют на корректность программы,

НО

от них сильно зависит читаемость!

Плохо:

`true_false`

`d`

`play_helper`

`my_int`

`l, I, 0`

Хорошо:

`rolled_a_one`

`dice`

`take_turn`

`num_rolls`

`k, i, m`

Имена должны отражать смысл или назначение связанных с ними значений.

Тип значения, связанного с именем, лучше описывать в докстринге функции.

Имена функций обычно отражают их действие (`print`), их поведение (`triple`), или возвращаемое значение (`abs`).

Кто заслуживает собственное имя?

Причины добавления нового имени

Повторение составных выражений:

```
if sqrt(square(a) + square(b)) > 1:  
    x = x + sqrt(square(a) + square(b))
```



```
hypotenuse = sqrt(square(a) + square(b))  
if hypotenuse > 1:  
    x = x + hypotenuse
```

**ПРАКТИЧЕСКИЕ
РЕКОМЕНДАЦИИ**

Смысловые части сложных выражений:

```
x = (-b + sqrt(square(b) - 4 * a * c)) / (2 * a)
```



```
discriminant = sqrt(square(b) - 4 * a * c)  
x = (-b + discriminant) / (2 * a)
```

Советы по именованию:

- Имена могут быть длинными, если они помогают документировать код:

```
average_age = average(age, students)
```

лучше, чем

```
# Вычисляем средний возраст студентов  
aa = avg(a, st)
```

- Имена могут быть короткими, если они связаны с очевидными величинами: счётчики, произвольные функции, аргументы математических действий и так далее.

n, k, i – обычно целые

x, y, z – обычно действительные

f, g, h – обычно функции

Тестирование

Разработка основанная на тестировании (TDD)

Пиши тест для функции до написания самой функции.

Тест прояснит поведение, область определения и область значений функции.

Тесты помогают отыскать «хитрые» граничные случаи.

Разрабатывай небольшими шагами и тестируй результат каждого шага.

Не полагайся на код, который не протестирован.

Всегда запускай все тесты при внесении изменений.

(Пример)

Дополнительная идея: запускай код в интерактивном режиме.

Не бойся экспериментировать с написанной функцией.

Интерактивные сессии могут превращаться в доктесты. Просто скопируй и вставь.

Декораторы

Декораторы функций

(Пример)

Декоратор
функции

```
@trace1  
def triple(x):  
    return 3 * x
```

Декорированная
функция

является идентичной с

Почему не делать
так?

```
def triple(x):  
    return 3 * x  
triple = trace1(triple)
```

Примеры задач

Представь себя «пайтоном» (ПСП)

Функция «print» возвращает «None». При вызове она также выводит на экран аргументы (разделенные пробелами).

```
from operator import add, mul
def square(x):
    return mul(x, x)
```

Функция, принимающая любой аргумент и возвращающая функцию, которая возвращает этот аргумент

```
def delay(arg):
    print('задержка')
    def g():
        return arg
    return g
```

Имена во вложенной инструкции «def» могут быть определены во внешней

Выражение	Значение	Интерактивный вывод
5	5	5
print(5)	None	5
print(<u>print(5)</u>)	None	5 None
<u>delay(delay)</u> (6)	6	задержка задержка 6
print(delay(print))(<u>4</u>)	None	задержка 4 None

Представь себя «пайтоном» (ПСП)

Функция «print» возвращает «None». При вызове она также выводит на экран аргументы (разделенные пробелами).

```
from operator import add, mul
def square(x):
    return mul(x, x)
```

Функция, которая всегда возвращает тождественную функцию

```
def pirate(arggg):
    print('йо-хо-хо')
    def plunder(arggg):
        return arggg
    return plunder
```

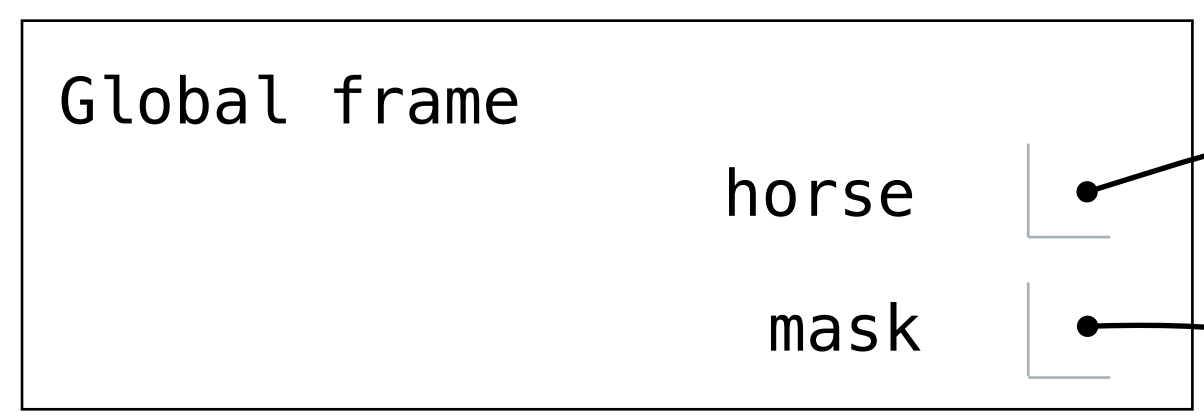
Выражение	Значение	Интерактивный вывод
<u>add(pirate(3)(square)(4), 1)</u>	17	йо-хо-хо 17
<u>func square(x)</u>		
<u>16</u>		
<u>pirate(pirate(pirate))(5)(7)</u>	Error	йо-хо-хо йо-хо-хо Error
<u>Тождественная функция</u>		
<u>5</u>		

Значение связанное с именем берётся из наиболее свежего фрейма текущего окружения, в котором оно определено.

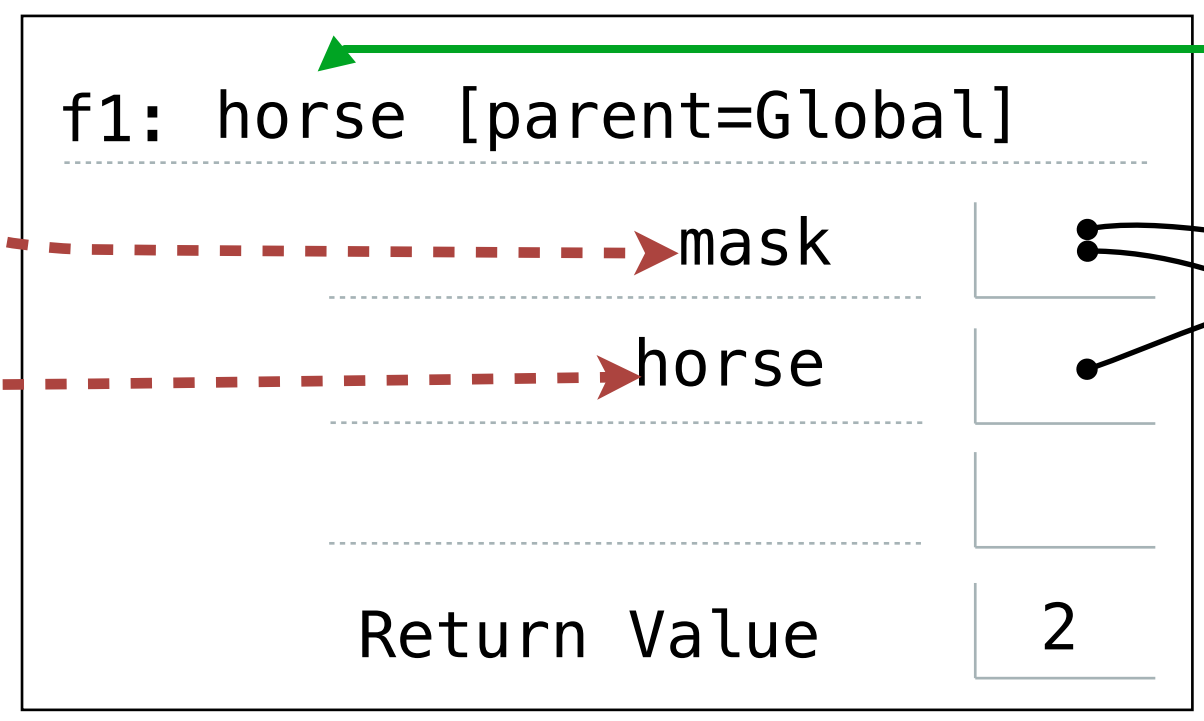
```
def horse(mask):  
    horse = mask  
    def mask(horse):  
        return horse  
    return horse(mask)
```

```
mask = lambda horse: horse(2)
```

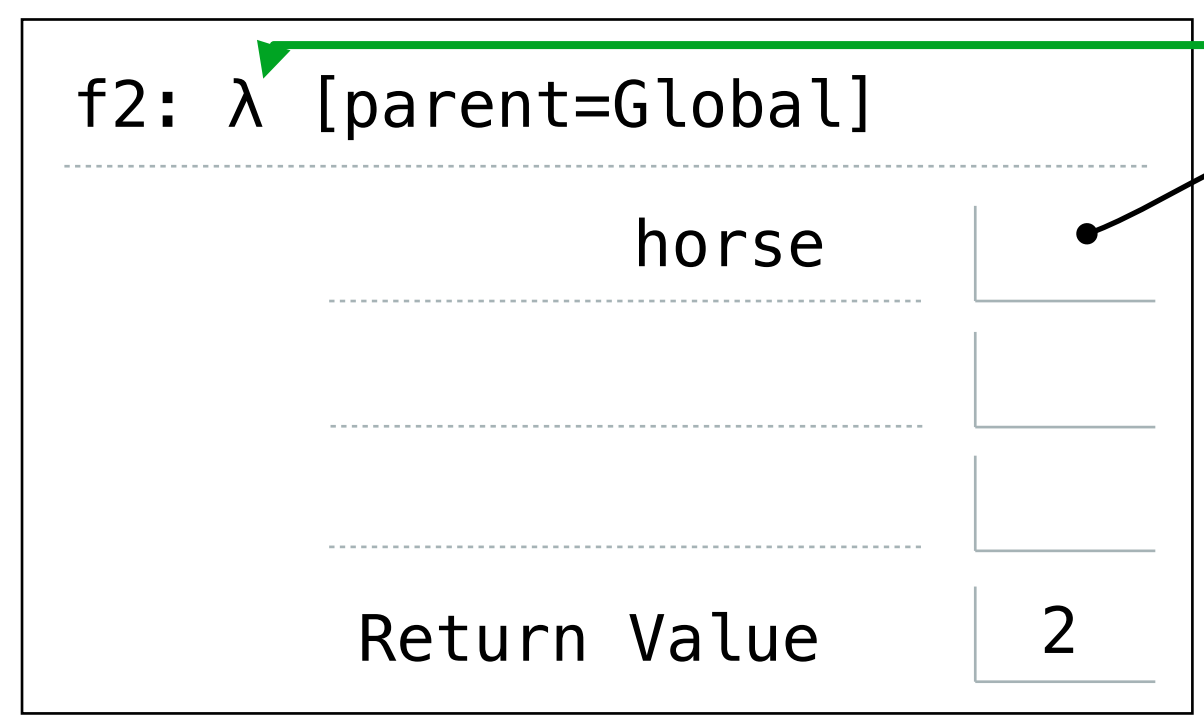
```
horse(mask)
```



func horse(mask) [parent=Global]



func λ(horse) [parent=Global]



func mask(horse) [parent=f1]

