Functional Abstraction

Announcements

Sign up for tutorials here!

Hog, Homework 1, and Lab 1 have been released

Hog checkpoint is due this Friday, 6/30. The entire project is due next Thursday 7/6, you can submit 1 day early, Wednesday 7/5, for a bonus point.

If you see a 0/1 for lab, don’t panic, it takes time to update. Please see this Ed post

Regular OH schedule this week

Instructor OH starts this week

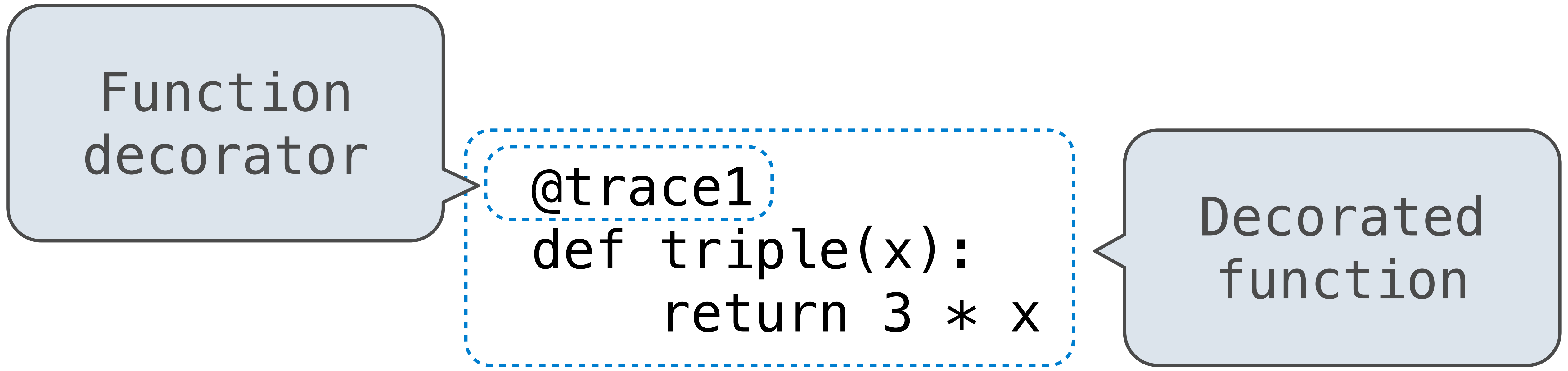
Advising OH starts this week

Sections are finalized on 6/30. No section switches after this point

Decorators

Function Decorators

(Demo)

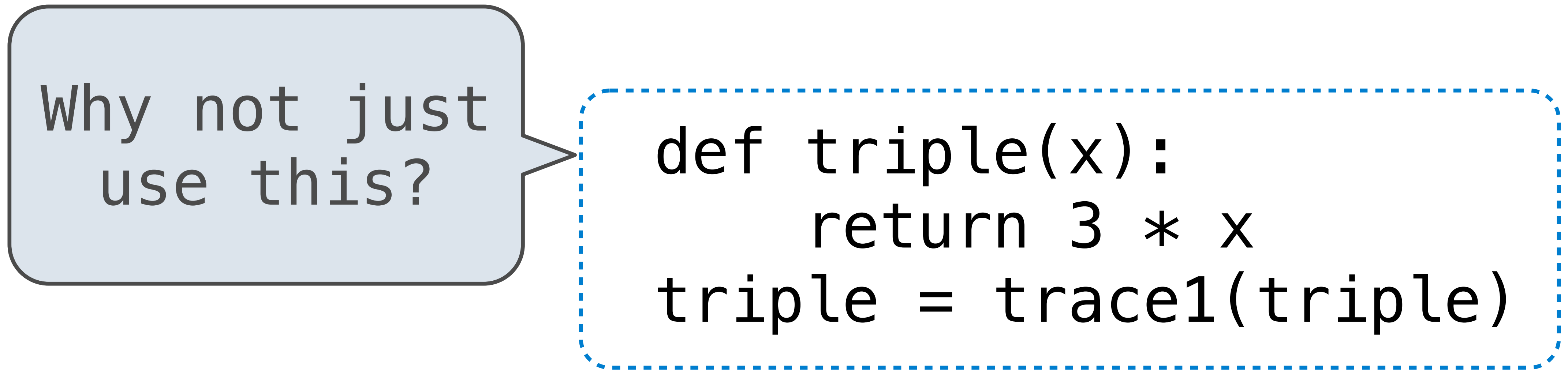


Function decorator

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

Decorated function

is identical to



Why not just use this?

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

Return

Return Statements

A return statement completes the evaluation of a call expression and provides its value:

f(x) for user-defined function f: switch to a new environment; execute f's body

return statement within f: switch back to the previous environment; f(x) now has a value

Only one return statement is ever executed while executing the body of a function

```
def end(n, d):
    """Print the final digits of N in reverse order until D is found.
    Assume N is non-negative

    >>> end(34567, 5)
    7
    6
    5
    while n > 0:
        last, n = n % 10, n // 10
        print(last)
        if d == last:
            return None
```

(Demo)

Abstraction

Functional Abstractions

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

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

What does sum_squares need to know about square?

- Square takes one argument. **Yes**
- Square has the intrinsic name square. **No**
- Square computes the square of a number. **Yes**
- Square computes the square by calling mul. **No**

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

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

If the name "square" were bound to a built-in function, sum_squares would still work identically.

Choosing Names

Names typically don’t matter for correctness

but

they matter a lot for composition

From:	To:
true_false	rolled_a_one
d	dice
play_helper	take_turn
my_int	num_rolls
l, I, O	k, i, m

Names should convey the meaning or purpose of the values to which they are bound.

The type of value bound to the name is best documented in a function's docstring.

Function names typically convey their effect (**print**), their behavior (**triple**), or the value returned (**abs**).

Which Values Deserve a Name

Reasons to add a new name

Repeated compound expressions:

```
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
```

Meaningful parts of complex expressions:

```
x1 = (-b + sqrt(square(b) - 4 * a * c)) / (2 * a)
```

```
discriminant = square(b) - 4 * a * c
x1 = (-b + sqrt(discriminant)) / (2 * a)
```

PRACTICAL GUIDELINES

More Naming Tips

- Names can be long if they help document your code:

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

  is preferable to

```
# Compute average age of students
aa = avg(a, st)
```

- Names can be short if they represent generic quantities: counts, arbitrary functions, arguments to mathematical operations, etc.

```
n, k, i - Usually integers
x, y, z - Usually real numbers
f, g, h - Usually functions
```

Break

Errors & Tracebacks

Taxonomy of Errors

Syntax Errors

Detected by the Python interpreter (or editor) before the program executes

Runtime Errors

Detected by the Python interpreter while the program executes

Logic & Behavior Errors

Not detected by the Python interpreter; what tests are for

(Demo)

Common Bugs

NameError

Spelling

Hello != hello != helo

SyntaxError

Missing parenthesis, Missing close quotes (EOL)

Logic & Behavior Errors

= vs ==

Infinite loops

Off by 1 errors

```
i = 0
while i < 10:
    print(i)
```

Common Bugs

IndentationError

Improper indentation

```
def f(x):
    print(x)
   return(x)
```

TypeError

Invalid types for an operator

Using non-function objects in a function call

Passing an incorrect number of arguments to a function

IndexError

Index a sequence with a number that exceeds the size of the sequence (preview to next week)

Debugging

(Demo)

Debugging Strategies and Techniques

Traceback messages

Running Doctests + writing your own tests

Using print statements (DEBUG: for okpy)

Interactive debugging

PythonTutor

Assert statements

Implementing Functions

Implementing a Function

```
def remove(n, digit):
    """Retu  n all  digits of non-negative N
         are          IT, for some
                      IT less than 10.

    >>> remove(231, 3)
    21
    >>> remove(243132, 2)
    4313
    """
    kept, digits = 0, 0

    while ______________n > 0______________:

        n, last = n // 10, n % 10

        if ______last != digit______________:

            kept = ~~10*~~kept + last*10**digits

            digits = ____digits + 1____________

    return ________kept____________________
```

231

4

231

```
   1        1
+ 20     + 30
         + 200
----     -----
  21      231
```

Read the description

Verify the examples & pick a simple one

Read the template

Implement without the template, then change your implementation to match the template.
OR
If the template is helpful, use it.

Annotate names with values from your chosen example

Write code to compute the result

Did you really return the right thing?

Check your solution with the other examples

Implementing a Function

```
def remove(n, digit):
    """Return all digits of non-negative N
       that are not DIGIT, for some
       non-negative DIGIT less than 10.

    >>> remove(231, 3)
    21
    >>> remove(243132, 2)
    4313
    """
    kept, digits = 0, 0

    while n > 0:

        n, last = n // 10, n % 10

        if last != digit:

            kept = kept/10 + last

            digits = digits + 1

    return round(kept * 10 ** (digits-1))
```

231

3

21

Read the description

Verify the examples & pick a simple one

Read the template

Implement without the template, then change your implementation to match the template.
OR
If the template is helpful, use it.

Annotate names with values from your chosen example

Write code to compute the result

Did you really return the right thing?

Check your solution with the other examples