爱扑bug的熊

SICP

God's Programming Book

Lecture-06 Design

cc BY NC SA

爱扑bug的熊

Design

Slides Adapted from cs61a of UC Berkeley

cc BY NC SA

Abstraction

cc BY NC SA

爱扑bug的熊

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.

爱扑bug的熊

Functional Abstractions

```
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.

cc BY NC SA

Naming Tips

cc BY NC SA

爱扑bug的熊

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`
`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**).

CC BY NC SA

爱扑bug的熊

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)
```

cc BY NC SA

爱扑bug的熊

Which Values Deserve a Name

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
```

cc BY NC SA

Thanks for Listening

CC BY NC SA