Lecture 4: Environment Diagrams

Brian Hou
June 21, 2016

Announcements

- Homework 1 is due Sunday 6/26
- Project 1 is released, due Thursday 6/30
  - Earn 1 EC point for completing it by Wednesday 6/29
- Go to discussion today! Each discussion is worth two *exam recovery points*
- Ask questions during lecture on Piazza!

Roadmap

Introduction

Functions

Data

Mutability

Objects

Interpretation

Paradigms

Applications

- This week (Introduction), the goals are:
  - To learn the fundamentals of programming
  - To become comfortable with Python

Abstraction

Abstraction

"The essence of abstraction is preserving information that is relevant in a given context, and forgetting information that is irrelevant in that context."

— John V. Guttag, *Introduction to Computation and Programming Using Python*

Discussion Question 1

n^2

$(n+1)^2$

$2 \cdot (n+1)$

$n^2 + 1$

$n \cdot (n+1)$

What does pyramid compute?

```python
def pyramid(n):
    a, b, total = 0, n, 0
    while b:
        a, b = a+1, b-1
        total = total + a + b
    return total
```

pyramid(4)

a   b

Tools for abstraction

- Assignment is a simple form of abstraction: bind names to values
- Function definition is a more powerful form of abstraction: bind names to a series of computations
- *Functional abstraction* is the idea that we can call functions without thinking about how the function works
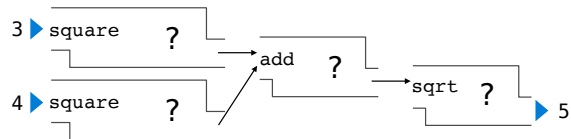
Miscellaneous Python features   (demo)

- Operators
- Multiple return values
- Docstrings
- Doctests
- Default arguments

Environment Diagrams

Lists and **for** Loops   (demo)

```python
s = [3, 1, 4, 1, 5, 9]
def max_difference(s):
    smallest = s[0]
    largest = s[0]
    for elem in s:
        if elem < smallest:
            smallest = elem
        if elem > largest:
            largest = elem
    return largest - smallest
max_difference(s)
```

Functions and **while** loops   (demo)

```python
x = 2
def repeated(f, n, x):
    while n > 0:
        x = f(x)
        n -= 1
    return x
def square(x):
    return x * x
repeated(square, x, 3)
```

Lambda Expressions

Lambda Expressions

```
>>> x = 10
```
An expression: this one evaluates to a number

```
>>> square = x * x
```
Also an expression: evaluates to a function

```
>>> square = lambda x: x * x
```
Important: No "return" keyword!

A function
with parameter x
that returns the value of "x * x"

```
>>> square(4)
16
```
Must be a single expression

Lambda expressions in Python cannot contain statements at all!

Lambda expressions aren't common in Python, but important in general

---

lambda   (demo)

```
x = 2
def repeated(f, n, x):
    while n > 0:
        x = f(x)
        n -= 1
    return x
def square(x):
    return x * x
repeated(square, x, 3)
```

```
x = 2
def repeated(f, n, x):
    while n > 0:
        x = f(x)
        n -= 1
    return x
square = lambda x: x * x
repeated(square, x, 3)
```

---

lambda