Announcements

• hw1 is due tomorrow at 11:59PM
  • Have to submit through your account
  • Your TA will go over homework submission in lab

• Project1 is out! Find a partner if you haven’t already.
  • Will have all the tools you need to complete the project by the end of lecture today
Let’s recap...
Looking up names

Procedure for looking up a name from inside a function (v. 1):

1. Look it up in the local frame
2. If not in local frame, look it up in the global frame
3. If in neither frame, generate error
Life cycle of a user-defined function

Def statement:
- Name: square(x):
- Return expression: return mul(x, x)
- Body (return statement)

What happens?
- Function created
- Name bound

Call expression:
- Operator: square
- Function: func square(x)
- Operand: 2+2
- Argument: 4

Op's evaluated
- Function called with argument(s)

Calling/Applying:
- Argument: 4
- Signature: square(x):
- Return value: 16

New frame!
- Params bound
- Body executed
Multiple environments in one diagram

Every expression is evaluated in the context of an environment.

```
from operator import mul

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

square(square(3))
```

Demo: [http://goo.gl/Nf7Lw](http://goo.gl/Nf7Lw)

“mul” is not found
Python Feature Demonstration

- Multiple Assignment
- Multiple Return Values
- Docstrings
- Doctests
- Default Arguments
def absolute_value(x):
    """Return the absolute value of x."""
    if x > 0:
        return x
    elif x == 0:
        return 0
    else:
        return -x

Boolean Contexts

False values in Python: False, 0, "", None

True values in Python: Anything else (True)

Read Section 1.5.4!
Keywords: “and” “or”

- The keywords “and” and “or” are useful for combining values in a boolean context
- and returns a true value if all expressions are true in a boolean context
  - (5 > 3) and (1 + 1 == 2) will return True
- or returns a true value if any expression is true in a boolean context
  - (1 > 5) or (400 < 10) or (2 == 4 – 2) will return True
- But it’s not quite that simple...
“Short-circuiting”

- The keyword “\(\text{and}\)” will return the first expression that is False in a boolean context
  - If there are no expressions that are False, then the last value in the statement is returned
- The keyword “\(\text{or}\)” will return the first expression that is True in a boolean context
  - If there are no expressions that are True, then the last value in the statement is returned

```python
>>> True and 5
5
>>> True or (5 / 0)
True
```
Interpreter session
Break
A statement is executed by the interpreter to perform an action.

Types of statements we have seen so far:

- An assignment
  
  ```python
  radius = 10
  ```

- A function definition
  
  ```python
  def square(x):
      return x * x
  ```

- Returns, imports, assertions
A function definition is a *compound statement*.

Compound statements:

```
<header>:
  <statement>
  <statement>
  ...
<bracket>
<separating	
  header>:
  <statement>
  <statement>
  ...
```

- **Statement**
- **Clause**
- **Suite**

The first header determines a statement’s type.

The header of a clause “controls” the suite that follows.
Compound Statements

Compound statements:

<header>:
  <statement>
  <statement>
  ...

< separating header >:
  <statement>
  <statement>
  ...

A suite is a sequence of statements

To “execute” a suite means to execute its sequence of statements, in order

Execution rule for a sequence of statements:

1. Execute the first
2. Unless directed otherwise, execute the rest
Conditional Statements

```python
def absolute_value(x):
    """Return the absolute value of x."""
    if x > 0:
        return x
    elif x == 0:
        return 0
    else:
        return -x
```

Execution rule for conditional statements:

Each clause is considered in order.
1. Evaluate the header's expression.
2. If it is a true value, execute the suite & skip the remaining clauses.
Local Assignment

```python
def percent_difference(x, y):
    difference = abs(x-y)
    return 100 * difference / x

diff = percent_difference(40, 50)
```

Execution rule for assignment statements:

1. Evaluate all expressions right of =, from left to right.
2. Bind the names on the left to the resulting values in the first frame of the current environment.
**Iteration**

1. Evaluate the header’s expression.
2. If it is a true value, execute the (whole) suite, then return to step 1.

Example: [http://goo.gl/mk7Sc](http://goo.gl/mk7Sc)
Break
Locally Defined Functions
Functions can be defined inside other functions

What happens when a def is executed?
1. Create a function value with the given signature and body
2. Bind the given name to that value in the current frame

The name can then be used to call the function.

def sum_of_squares(n):
    """Sum of the squares of the integers 1 to n""

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

total, k = 0, 1
while k <= n:
    total, k = total + square(k), k + 1
return total
Locally Defined Functions
The inner definition is executed each time the outer function is called

```python
1 from operator import mul
2 def square_inside():
3     def square(x):
4         return mul(x, x)
5     square_inside()
6     square_inside()
```
Higher-Order Functions

Functions are first-class: they can be manipulated as values in Python

Higher-order function: a function that takes a function as an argument value or returns a function as a return value

Higher order functions:
- Express general methods of computation
- Remove repetition from programs
- Separate concerns among functions
The Art of the Function

- Give each function exactly one job
- Don’t reapeat yourself (DRY).
- Don’t reapeat yourself (DRY).
- Define functions generally
- Proj1 has a composition score! Adhere to these guidelines
Generalizing Patterns with Parameters

Regular geometric shapes relate length and area.

Shape:

\[ r \]

Area:

1. \( r^2 \)
2. \( \pi r^2 \)
3. \( \frac{3\sqrt{3}}{2} r^2 \)

Finding common structure allows for shared implementation.
Interpreter session
Generalizing Over Computational Processes

The common structure among functions may itself be a computational process, rather than a number.

\[
\sum_{k=1}^{5} k = 1 + 2 + 3 + 4 + 5 = 15
\]

\[
\sum_{k=1}^{5} k^3 = 1^3 + 2^3 + 3^3 + 4^3 + 5^3 = 225
\]

\[
\sum_{k=1}^{5} \frac{8}{(4k - 3) \cdot (4k - 1)} = \frac{8}{3} + \frac{8}{35} + \frac{8}{99} + \frac{8}{195} + \frac{8}{323} = 3.04
\]
Interpreter session
Functions as Arguments
Function values can be passed as arguments

```python
def cube(k):
    return pow(k, 3)

def summation(n, term):
    '''Sum the first n terms of a sequence.
    >>> summation(5, cube)
    225'''
    total, k = 0, 1
    while k <= n:
        total, k = total + term(k), k + 1
    return total

>>> summation(5, cube)
225

0 + 1^3 + 2^3 + 3^3 + 4^3 + 5^5
```

Function of a single argument (not called term)

A formal parameter that will be bound to a function

The cube function is passed as an argument value

The function bound to term gets called here
Function Values as Parameters

Parameters can be bound to function values

Example: [http://goo.gl/e4YBH](http://goo.gl/e4YBH)
That’s it for today

- This is all I wanted to get through for today, but if we have time left, we can go to the next slides
Functions as Return Values
Locally defined functions can be returned
They have access to the frame in which they are defined

A function that returns a function

```python
def make_adder(n):
    """Return a function that adds n to its argument."
    
    >>> add_three = make_adder(3)
    >>> add_three(4)
    7
    """

def adder(k):
    return add(n, k)
return adder
```

The name add_three is bound to a function

A local def statement

Can refer to names in the enclosing function
Call Expressions as Operators

```
def make_adder(n):
    def adder(k):
        return add(n, k)
    return adder

make_adder(1)(2)
```
Interpreter Session

• This concept usually trips some students up
• Let’s see it in the interpreter
Higher-Order Functions

Functions are first-class: they can be manipulated as values in Python

Higher-order function: a function that takes a function as an argument value or returns a function as a return value

Higher order functions:
- Express general methods of computation
- Remove repetition from programs
- Separate concerns among functions
Tomorrow…

• How do higher order functions look in Environment diagrams?
• Homework 1 is due
• Office hours today, see website