Structure and Interpretation of Computer Programs with Python

Lesson 2

-Presented By: Sean Li

Quick Review

Call Expression:

- Components:

Quick Review

Call Expression:

- Components: Operator & Operand
- Evaluation Process:

Quick Review

Call Expression:

- Components: Operator & Operand
- Evaluation Process:
  - From left to right
  - First evaluates the operator, then the operand
  - Apply the function that is the value of the operator to the arguments that are the values of the operands

Names

Expressions

2

3*5

F(x)

divide(1, 2)

Types of Expressions

Primitive Expression: 2, add, "hello"

Number or Numeral Name String

Call Expression: add(1, 2)

Assignment (not that assignment)

Any difference?

```
a = 1
a == 1
```

Assignment (not that assignment)

Any difference?

```
a = 1
a == 1
```

assignment

equality operator

Assignment (not that assignment)

Any difference?

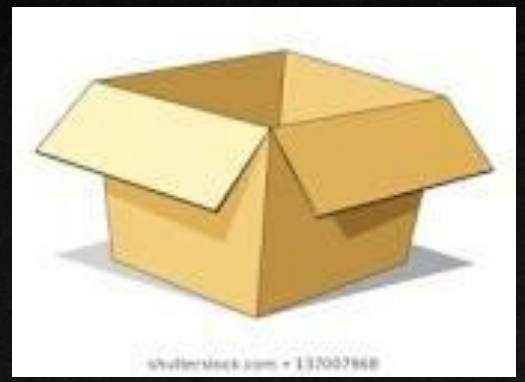
a = 1 assignment

a == 1 equality operator

Assignment is a simple means of abstraction: binds names to values

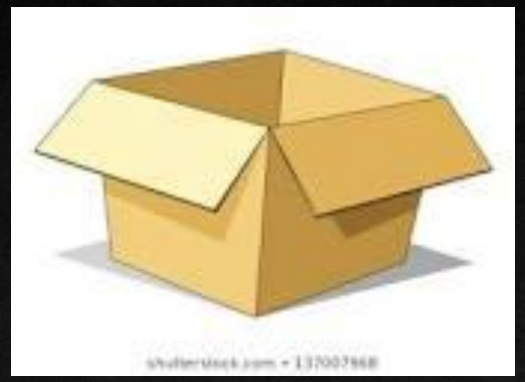
Assignment (not that assignment)

a

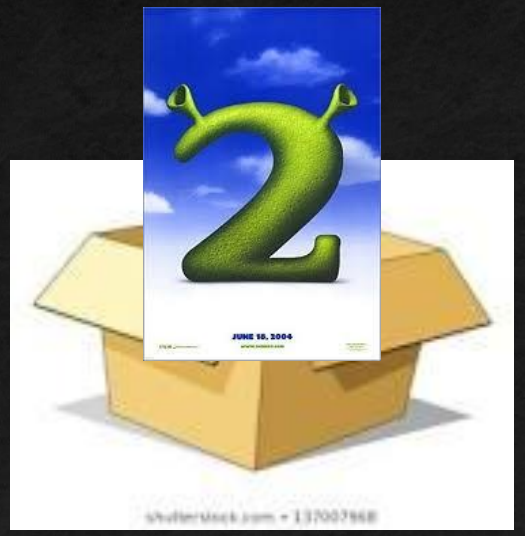


Assignment (not that assignment)

a

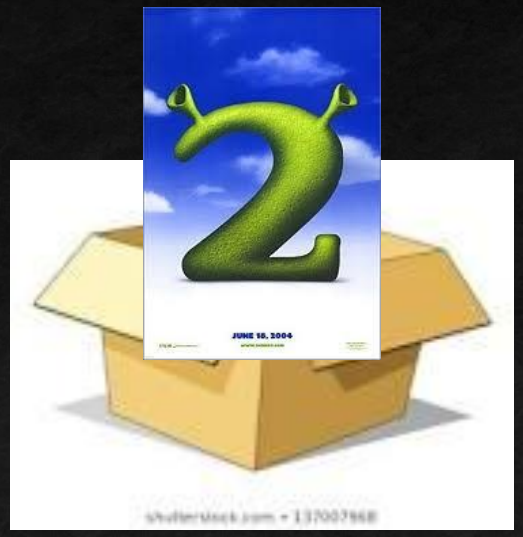


a = 2



Assignment (not that assignment)

a?



demo

Define a Function:

Function definition is a more powerful means of abstraction: binds names to expressions

```
def <name>(<formal parameters>):
    return <return expression>
```

Define a Function:

Function name

Function signature

```
def <name>(<formal parameters>):
    return <return expression>
```

Function body

Define a Function:

Function name: the name of the function

Function signature: the arguments that the function takes

```
def <name>(<formal parameters>):
    return <return expression>
```

Function body: defines the computation performed when the function is applied

Execution procedure for def statements:

1. Create a function with signature <name>(<parameters>)
2. Set the body of that function to be everything indented after the first line
3. Bind <name> to that function

```
def add(a, b):
  return a + b
```

Are they the same?

```
def add(a, b):
  return a + b

def add(a, b):
  result = a + b
  return result

def add(hello, world):
  return hello + world
```

What would python display?

```
def add(a, b):
  return a + b


add, a, b
```

A variable only lives within the context in which it was created

```
def add(a, b):
  return a + b

add, a, b
```

Example:

At home, you are the son of your parents.

At school, you are a student.

You can't be a son at school or a student at home.

Exercise:

Define a function that prints out whatever string gets passed in the argument.

Exercise:

Using def statement, define your own calculator, that has the function of plus minus multiply and divide.

Challenge:

Write a function that takes three *positive* numbers and returns the sum of the squares of the two smallest numbers. **Use only a single line for the body of the function.**

```
def two_of_three(x, y, z):
    """Return a*a + b*b, where a and b are the two smallest members of the
    positive numbers x, y, and z.

    >>> two_of_three(1, 2, 3)
    5
    >>> two_of_three(5, 3, 1)
    10
    >>> two_of_three(10, 2, 8)
    68
    >>> two_of_three(5, 5, 5)
    50
    >>> # check that your code consists of nothing but an expression (this docstring)
    >>> # a return statement
    >>> import inspect, ast
    >>> [type(x).__name__ for x in ast.parse(inspect.getsource(two_of_three)).body[0].body]
    ['Expr', 'Return']
    """
    return _____
```

Hint: Consider using the `max` or `min` function:

```
>>> max(1, 2, 3)
3
>>> min(-1, -2, -3)
-3
```