More About Functions

Functions

A ***function*** is a named subprogram.

- Used by calling its name.
- ***Encapsulation*** – wrapping code into a unit.
- ***Generalization*** – parameters allow more general solutions.

Parameters and arguments

A variable in the function ***heading*** is called a *parameter*.

A parameter is a place holder for the *argument* passed when the function is called.

Parameters and arguments

A variable in the function ***heading*** is called a *parameter*.

A parameter is a place holder for the *argument* passed when the function is called.

Parameters and arguments

A variable in the function ***heading*** is called a *parameter*.

A parameter is a place holder for the *argument* passed when the function is called.

A parameter is always a *variable*.

An argument is always a *value*.

Parameters and arguments

Consider this very simple function.

```
def foo(x):
  print x
  print x
```

What does this do?

This function simply prints `x` twice.

Parameters and arguments

What does this do?

```
>>> def foo(x):
      print x
      print x
>>> foo(1823)
1823
1823
```


Parameters and arguments

Consider this very simple function.

```
def foo(x):
  print x
  print x
```

What does this do?

Parameters and arguments

What does this do?

```
>>> def foo(x):
      print x
      print x
>>> foo(1823)
```

Parameters and arguments

What does this do?

```
>>> def foo(x):
      print x
      print x
>>> foo('Trinity')
Trinity
Trinity
```


Parameters and arguments

What does this do?

```
>>> def foo(x):
      print x
      print x
>>> foo('Trinity')
```


Parameters and arguments

What does this do?

```
>>> def foo(x):
      print x
      print x
>>> n = 3
>>> foo(n+4)
7
7
```


Parameters and arguments

What does this do?

```
>>> def foo(x):
      print x
      print x
>>> n = 3
>>> foo(n+4)
```

Parameters and arguments

What does this do?

```
>>> def foo(x):
      print x
      print x
>>> foo('Trin'*2)
TrinTrin
TrinTrin
```


Parameters and arguments

Consider this very simple function.

```
def goo(x, y):
  print '1st value:', x
  print '2nd value:', y
```

What does this do?

Parameters and arguments

What does this do?

```
>>> def foo(x):
      print x
      print x
>>> foo('Trin'*2)
```


Parameters and arguments

If a function has *n* parameters, a caller must pass *n* arguments.

Arguments are assigned to parameters according to the order alone.

Parameters and arguments

What does this do?

```
>>> def goo(x, y):
      print '1st value:', x
      print '2nd value:', y
>>> goo(3, 4)
```


Parameters and arguments

Consider this very simple function.

```
def goo(x, y):
  print '1st value:', x
  print '2nd value:', y
```

What does this do?

This function simply prints the values of the parameters `x` and `y`.

Parameters and arguments

What does this do?

```
>>> def goo(x, y):
      print '1st value:', x
      print '2nd value:', y
>>> a = 5
>>> b = 6
>>> goo(a, b)
```


Parameters and arguments

What does this do?

```
>>> def goo(x, y):
      print '1st value:', x
      print '2nd value:', y
>>> goo(3, 4)
1st value: 3
2nd value: 4
```

Parameters and arguments

What does this do?

```
>>> def goo(x, y):
      print '1st value:', x
      print '2nd value:', y
>>> x = 7
>>> y = 8
>>> goo(y, x)
```


Parameters and arguments

What does this do?

```
>>> def goo(x, y):
      print '1st value:', x
      print '2nd value:', y
>>> a = 5
>>> b = 6
>>> goo(a, b)
1st value: 5
2nd value: 6
```


Local and Global Variables

```
>>> def goo(x, y):      # local x, y
      x = x + 1
      y = y + 1
      print 'x =', x    # print local
      print 'y =', y
>>> x = 7                # global x, y
>>> y = 8
>>> goo(y, x)
print 'x =', x         # print global
print 'y =', y
x = 9                 # local values
y = 8
x = 7                 # global values
y = 8
```


Parameters and arguments

What does this do?

```
>>> def goo(x, y):
      print '1st value:', x
      print '2nd value:', y
>>> x = 7   # Not the same as parameter x
>>> y = 8   # Not the same as parameter y
>>> goo(y, x)
1st value: 8
2nd value: 7
```

Parameters and arguments

What does this do?

```
>>> def hoo(x, y):
      n = x
      print '1st value:', n
      print '2nd value:', y
>>> n = 1
>>> y = 2
>>> hoo(y, n)
```


Variable & Parameter Scope

Variables and parameters defined inside a function are *local*: they can only be used inside the function.

Parameters and arguments

Some scope errors

```
>>> def hoo(x, y):
      n = x
      print '1st value:', n
      print '2nd value:', y

>>> hoo(x, y)   # Error: x, y undefined
>>> print n     # Error: n undefined
>>> print x     # Error: x undefined
```


Parameters and arguments

What does this do?

```
>>> def hoo(x, y):
      n = x
      print '1st value:', n
      print '2nd value:', y
>>> n = 1
>>> y = 2
>>> hoo(y, n)
1st value: 2
2nd value: 1
```

Global vs Local Variables

```
a = 10
b = 11
def goo(x, y):
    x = x + 1
    y = y + 1
    print 'x =', x
    print 'y =', y
    print 'a =', a
    print 'b =', b
goo(a, b)
print 'a =', a
print 'b =', b
```

Output

```
x = 11
y = 12
a = 10
b = 11
a = 10
b = 11
```