FUNCTIONS AND PARAMETERS

CSSE 120 – Rose-Hulman Institute of Technology

Outline

- Review of topics for Exam #1
- Basic Functions - Math, Maple, Python
- Function definition and invocation mechanics
- printFactorial example
- printDistance exercise
- Preview: Functions that return a value
- Nested function calls and execution order
- Code-reading exercise
- Functions and Graphics (finish for homework)

Possible Topics for Exam 1(Tuesday)

- algorithm
- comment
- variable, assignment
- identifier, expression
- loop
  - definite (for)
  - counted (range function)
- phases of software development
- input, print
- import, math functions
- using functions
- int, float, long, conversion
- strings (basic operations)
- character codes (chr, ord)
- lists (concatenation, slices)
- reading, writing files
- formatted output
- using objects, graphics
- method *vs* function
- set window coordinates
- event-driven program

Why functions?

- A function allows us to group together several statements and give them a name by which they may be invoked.
- Abstraction (easier to remember the name than the code)
- Compact (avoid duplicate code)
- Flexibility(parameters allow variation)

Functions in different realms

- We compare the mechanisms for defining and invoking functions in three different settings:
- Standard Mathematical notation
- Maple
- Python

Functions in Mathematics

- Define a function:
  - $f(x) = x^2 - 5$

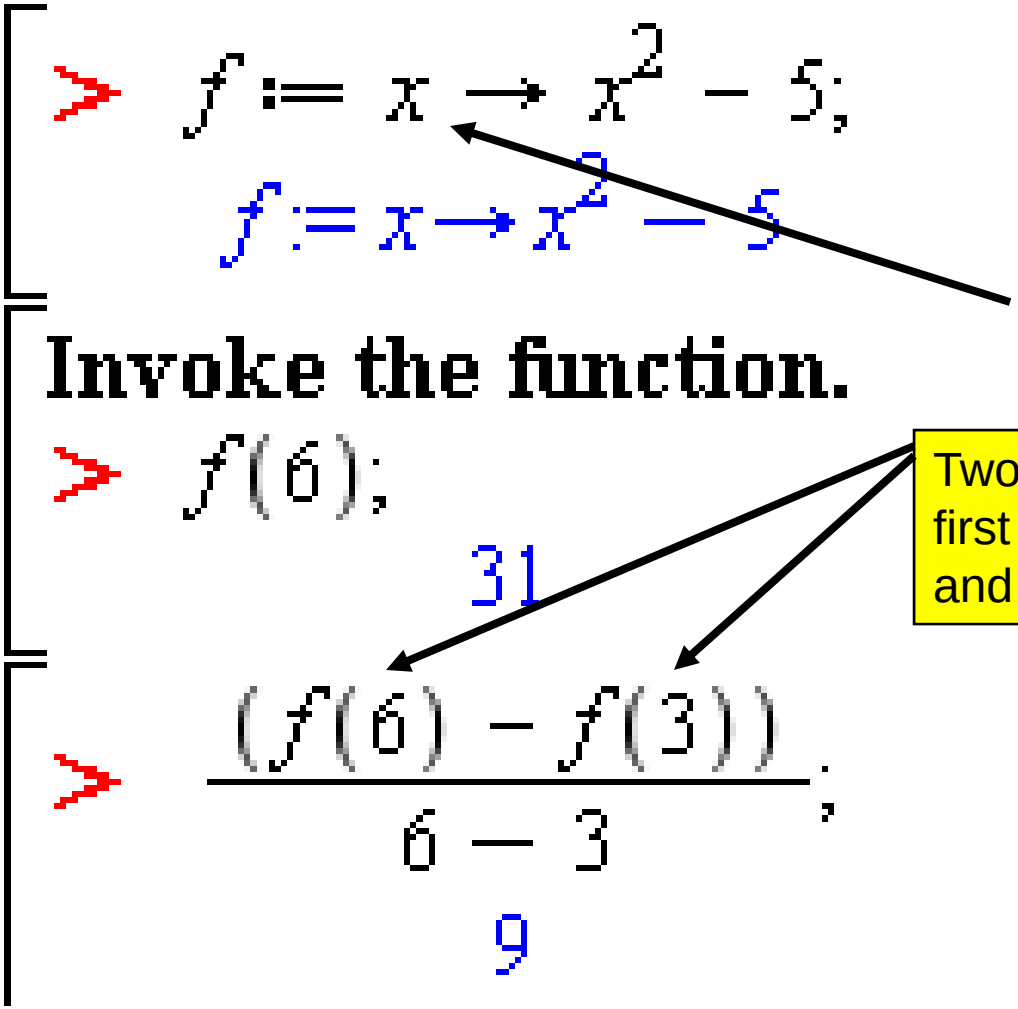
Formal Parameter. Used so that we have a name to use for the argument in the function's formula.

- Invoke (call) the function:
  - $\dfrac{f(6) - f(3)}{6 - 3}$

Two calls to function **f**. The first with actual parameter 6, and the second with 3.

- When the call f(6) is made, the actual parameter 6 is substituted for the formal parameter x, so that the value is $6^2 - 5$.

Functions in Maple



Formal Parameter. Used so that we have a name to use for the argument in the function's formula.

Two calls to function **f**. The first with actual parameter 6, and the second with 3.

Functions in Python

```
>>> def f(x):
        return x*x - 5

>>> f(6)
31
>>> (f(6) - f(3)) / (6 - 3)
9
>>>
```

Formal Parameter. Used so that we have a name to use for the argument in the function's formula.

Two calls to function **f**. The first with actual parameter 6, and the second with 3.

- In Mathematics, functions calculate a value.
- In Python we often define functions that instead *do something*, such as print some values.

Review: Parts of a Function Definition

```
>>> def hello():
        print "Hello"
        print "I'd like to complain about this parrot"
```

Defining a function called "hello"

Indenting tells interpreter that these lines are part of the hello function

Blank line tells interpreter that we're done defining the hello function

Review: Defining vs. Invoking

- Defining a function says what the function should do
- Invoking a function makes that happen
  - Parentheses tell interpreter to invoke the function

```
>>> hello()
Hello
I'd like to complain about this parrot
```

Review: Function with a Parameter

- def complain(complaint):

```
    print "Customer: I purchased this parrot not half " +
          "an hour ago from this very boutique"
    print "Owner: Oh yes, the Norwegian Blue. " +
         " What's wrong with it?"
    print "Customer:", complaint
```

- invocation:
  - complain("It's dead!")

When a function is invoked

When Python comes to a function call, it initiates a four-step process:

1. The calling program suspends execution at the point of the call.
2. The formal parameters of the function get assigned the values supplied by the actual parameters in the call.
3. The body of the function is executed.
4. Control returns to the point just after where the function was called.

printFactorial function

```
def printFactorial(n, formatWidth):
  formatString = "%"+str(formatWidth)+"d"
  product = 1
  for i in range(1, n+1):
      product = product * i
  print formatString % (product)
```

- Use this to compute and print a large factorial:
  - `printFactorial(15, 13):`
- Note that the substitution of actual parameters for formal parameters is done in order.
- Use `printFactorial` to create a table of factorials

Exercise – with a partner

- Write and test a `printDistance` function that has two Points as parameters, and prints the distance between them.
  - Don't forget `from graphics import Point`
- Test your function with the following code:
  - `printDistance(Point(5,6), Point(1, 9))`
  - `printDistance(Point(12, 17), Point(6, 8))`
- Recall how to get the x and y coordinates of a Point.
- Put your code file in the `printDistance` drop box, making sure that the title of your submission contains both partners' names.

If a Function Calls a Function …

```
def g(a,b):
    print a+b, a-b

def f(x, y):
    g(x, y)
    g(x+1, y-1)

f(10, 6)
```

- Trace what happens when the last line of this code executes

Optional parameters

- A python function may have some parameters that are optional.

```
>>> int("37")
37
>>> int("37", 10)
37
>>> int("37", 8) # specify base 8
31
```

We can declare a parameter to be optional by supplying a default value.

```
>>> def printDate(month, day, year=2007):
        print month, str(day)+",", year


>>> printDate("January", 4, 2006)
January 4, 2006
>>> printDate("January", 4)
January 4, 2007
```

Celsius → Fahrenheit

- A preview of Day 8: a function that returns a value.
- Define it

```
def celsiusToFahrenheit(celTemp):
    return 9.0/5 *celTemp + 32
```

- Call it

```
print "100 degrees C = %0.2f degrees F" % \
      (celsiusToFahrenheit(100))
```

- Output

```
100 degrees C = 212.00 degrees F
```

An exercise in code reading

- With a partner, read and try to understand the code that is on the back of the quiz paper.
- You can probably guess what the output will be. But how does it work.
- Figure that out, discuss it with your partner and answer quiz question 9.
- Optional Challenge Problem for later: try to write "There's a Hole in the Bottom of the Sea" in a similar style.
- When you are done, turn in your quiz and do the exercise from the next slide.

Graphic Pizza Function

- Write a function `drawPizza` whose parameters are
  - a GraphWin object
  - a Point object (the center of the circle)
  - the radius
  - the number of slices.
- Test your function with this code:
  - ```
    from graphics import *
    win = GraphWin(""600, 600)
    pizza(win, Point(400,400), 150, 12)
    ```
- In the homework, you will add `poly()` and `star()` functions. If you have time, begin them now.
    ```