



PLAN 396

Lecture 13

Dr. Hossen Asiful Mustafa

<https://hossenmustafa.buet.ac.bd>



Introduction

- We have seen several built-in functions
 - `len()`
 - `range()`
 - `random.randrange()`
- These are standard functions that are provided to you with Python
- But what if you want to make your own function?



Introduction

- Components

- Consist of functions, classes, modules and packages
- In most cases programs are a combination of programmer defined functions and classes with predefined ones

- Programmer defined functions

- Programs that perform specific tasks and execute at various points in the program



Functions

- Invoked by a function call
 - Specifies the function name and its arguments
- Arguments
 - Additional information the function needs to complete its task
- The return task
 - Information that the function returns to the source that invoked it for use elsewhere in the program



Module

- Contains function definitions and other elements
 - All of which are related in some way
- Calling a function
 - `functionName (argument1, argument2)`
- The `import` keyword is used to include a module
- Invoking functions from a module
 - Use the module name followed by the dot operator (`.`)
 - `moduleName.functionName(argument)`



Functions

- Module math Functions

```
Python 2.2b2 (#26, Nov 16 2001, 11:44:11) [MSC 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import math
>>> print math.sqrt( 900 )
30.0
>>> print math.sqrt( -900 )
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
ValueError: math domain error
```



Functions

Method	Description	Example
acos (<i>x</i>)	Trigonometric arc cosine of <i>X</i> (result in radians)	acos (1.0) is 0.0
asin (<i>x</i>)	Trigonometric arc sine of <i>X</i> (result in radians)	asin (0.0) is 0.0
atan (<i>x</i>)	Trigonometric arc tangent of <i>X</i> (result in radians)	atan (0.0) is 0.0
ceil (<i>x</i>)	Rounds <i>X</i> to the smallest integer not less than <i>X</i>	ceil (9.2) is 10.0 ceil (-9.8) is -9.0
cos (<i>x</i>)	Trigonometric cosine of <i>X</i> (<i>X</i> in radians)	cos (0.0) is 1.0
exp (<i>x</i>)	Exponential function e^x	exp (1.0) is 2.71828 exp (2.0) is 7.38906
fabs (<i>x</i>)	Absolute value of <i>X</i>	fabs (5.1) is 5.1 fabs (-5.1) is 5.1
floor (<i>x</i>)	Rounds <i>X</i> to the largest integer not greater than <i>X</i>	floor (9.2) is 9.0 floor (-9.8) is -10.0
fmod (<i>x</i> , <i>y</i>)	Remainder of <i>X</i> / <i>y</i> as a floating point number	fmod (9.8, 4.0) is 1.8



Functions

hypot(x, y)	hypotenuse of a triangle with sides of length x and y: $\sqrt{x^2 + y^2}$	hypot(3.0, 4.0) is 5.0
log(x)	Natural logarithm of x (base e)	log(2.718282) is 1.0 log(7.389056) is 2.0
log10(x)	Logarithm of x (base 10)	log10(10.0) is 1.0 log10(100.0) is 2.0
pow(x, y)	x raised to power y (x^y)	pow(2.0, 7.0) is 128.0 pow(9.0, .5) is 3.0
sin(x)	trigonometric sine of x (x in radians)	sin(0.0) is 0.0
sqrt(x)	square root of x	sqrt(900.0) is 30.0 sqrt(9.0) is 3.0
tan(x)	trigonometric tangent of x (x in radians)	tan(0.0) is 0.0



Creating Functions

○ Definitions

- Functions must be defined before they are used
- **`def functionName (paramList):`**
 - functionName is a valid identifier
 - paramList is a comma separated list of parameters received
 - The actions of the functions then follows
 - They should all be indented appropriately
 - The actions are also called the block or the function body



Creating Functions

- Documentation

- Python uses a “documentation string” for functions (aka. docstring)
 - A triple quoted string immediately following the function definition statement

- Example

```
def instructions()  
    """Display game instructions"""
```

Using Parameters and Return Values

function definition

```
def square( y ):
```

```
    return y * y
```

```
for x in range( 1, 11 ):
```

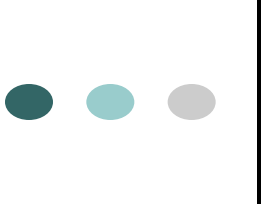
```
    print square( x )
```

This is a function definition, the function is called **square** and is passed the value **y**

The function returns the passed value multiplied by itself

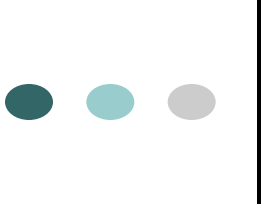
This calls the **square** function and passes it the value **x**

1 4 9 16 25 36 49 64 81 100



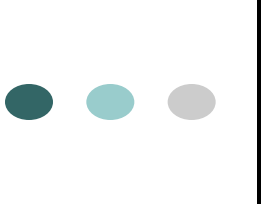
Keyword Arguments and Default Values

- Parameters in Python can be passed in many different ways and use different concepts
 - positional parameters
 - keyword arguments
 - named parameters
 - default values




Keyword Arguments and Default Values

- Positional parameters is a method of passing parameters to a function using the convention of “position” of the argument to the called function.
 - The first parameter gets the first value sent
 - The second parameter gets the second value sent
 - ...
- Example:
 - `def birthday(name, age)`
 - `birthday(“Sarah”, 4)`
 - `birthday(4, Sarah)`



Keyword Arguments and Default Values

- Default Function Arguments
 - Functions may commonly receive a particular value type
 - When this is true a default argument can be set
 - Must appear to the right of any other arguments
 - When omitted all value to the right must also be omitted
 - A default value can also be set
 - If passes a value then the default value is overridden

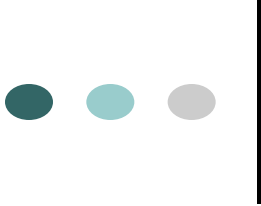


Keyword Arguments and Default Values

function definition with default arguments


```
def boxVolume( length = 1, width = 1, height = 1 ):
    return length * width * height
```

```
print "The default box volume is:", boxVolume()
print "\nThe volume of a box with length 10,"
print "width 1 and height 1 is:", boxVolume( 10 )
print "\nThe volume of a box with length 10,"
print "width 5 and height 1 is:", boxVolume( 10, 5 )
print "\nThe volume of a box with length 10,"
print "width 5 and height 2 is:", boxVolume
```



Keyword Arguments and Default Values

- Keyword arguments
 - Just as a programmer specifies default arguments keyword arguments can be specified as well
 - Allows parameter to be passed in any order so long as they are explicitly stated
 - Will set the values that were not passed to the default



Keyword Arguments and Default Values

```
def generateWebsite( name, url = "www.deitel.com", Flash = "no", CGI = "yes" ):
    print "Generating site requested by", name, "using url", url
```

```
    if Flash == "yes":
        print "Flash is enabled"
```

```
    if CGI == "yes":
        print "CGI scripts are enabled"
    print # prints a new line
```

```
generateWebsite( "Deitel" )
generateWebsite( "Deitel", Flash = "yes", url = "www.deitel.com/new" )
generateWebsite( CGI = "no", name = "Prentice Hall" )
```



Global Variables and Constants

- Rules for value retrieval
 - Based on namespace and scope
 - Namespaces store information about an identifier and a value to which it is bound
 - Three types
 - Local, global, and built-in
 - They are also checked by Python in the order listed above
- Local namespace
 - Contains values that were created in a block
 - Each function has a unique local namespace

Global Variables and Constants

```
x = 1 # global variable
```

This is a global variable and can be used by any function in the program

```
# alters the local variable x, shadows the global variable
```

```
def a():
```

```
    x = 25
```

```
    print "\nlocal x in a is", x, "after entering a"
```

```
    x += 1
```

```
    print "local x in a is", x, "before exiting a"
```

Has its own value of **x** therefore the global value is hidden

```
# alters the global variable x
```

```
def b():
```

```
    global x
```

```
    print "\nglobal x is", x, "on entering b"
```

```
    x *= 10
```

```
    print "global x is", x, "on exiting b"
```

Function **b** uses and modifies the value of the global **x**



Passing Lists to Functions

- Passing a list
 - To pass a list pass it without its brackets
 - This allows the entire list to be changed
 - Item in the list that are immutable (numbers or strings) cannot be changed by the function when passed individually



Passing Lists to Functions

```
def modifyList( aList ):
    for i in range( len( aList ) ):
        aList[ i ] *= 2
```

```
def modifyElement( element ):
    element *= 2
```

```
aList = [ 1, 2, 3, 4, 5 ]
```

```
print "Effects of passing entire list:"
print "The values of the original list are:"
```

```
for item in aList:
    print item,
    modifyList( aList )
```

```
print "\n\nThe values of the modified list are:"
```

```
for item in aList:
    print item,
```



Class Assignment

- Write a program named `classassignment16.py`
- The program should have 2 functions:
 - A function that converts temperature from Celsius to Fahrenheit
 - A function that converts temperature from Fahrenheit to Celsius
 - Test the functions by getting inputs from users and print outputs