

Functions

by Deborah R. Fowler



- Perform a task, then return control to your program (from where they were called)
- Functions allow us to break up big programs into smaller pieces



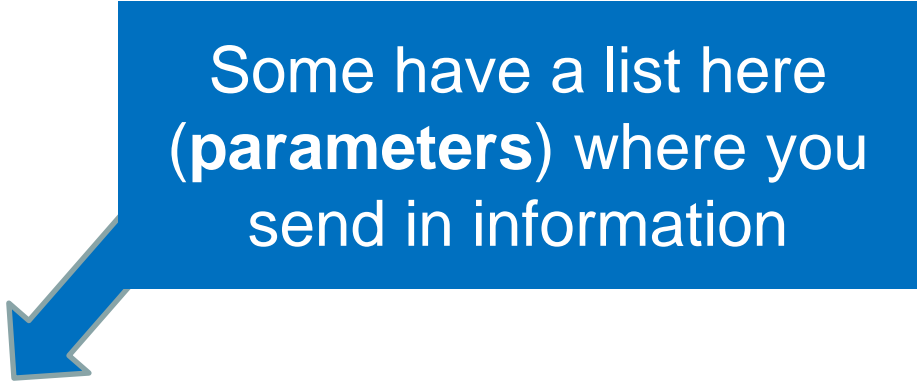
For example:

```
def myFunction():  
    // things defined here exist only here (code block)  
    // that is called encapsulation (it is a good thing)  
    // statements to do something go here  
    // called the body of the function  
    return value    // returns a value - optional
```

The code is executed when the name is called

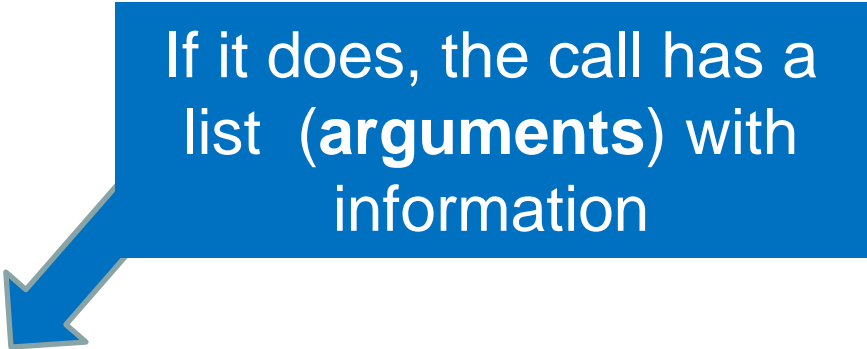
```
someVariable = myFunction()
```

Some have a list here
(**parameters**) where you
send in information



```
myFunction():  
    // myFunction does something  
    return value // return
```

If it does, the call has a
list (**arguments**) with
information



```
someVariable = myFunction()
```

File Edit Format Run Options Window Help

```
# Mystery Function example
```

```
def main():  
    num1 = 5  
    num2 = 7  
  
    result = mysteryFunction(num1, num2)  
    print "The number is " + str(result)
```

Called with num1 and num2 (5 and 7)

```
# Functions should also include a comment
```

```
def mysteryFunction(a, b):  
    returnValue = 1  
  
    if ( b < 0 ):  
        return 0  
    for i in range(0, b):  
        returnValue *= a  
    return returnValue
```

Two parameters a and b

Body of the function

```
|  
main()
```

File Edit Format Run Options Window Help

```
# Mystery Function example
```

```
def main():  
    num1 = 5  
    num2 = 7  
  
    result = mysteryFunction(num1, num2)  
    print "The number is " + str(result)
```

The number is 78125

```
# Functions should also include a comment
```

```
def mysteryFunction(a, b):  
    returnValue = 1  
  
    if ( b < 0 ):  
        return 0  
    for i in range(0, b):  
        returnValue *= a  
    return returnValue
```

So what does our
mysteryFunction do?

```
|  
main()
```



File Edit Format Run Options Window Help

```
# Mystery Function example
```

```
def main():  
    num1 = 5  
    num2 = 7  
  
    result = mysteryFunction(num1, num2)  
    print "The number is " + str(result)
```

```
# Functions should also include a comment
```

```
def mysteryFunction(a, b):  
    returnValue = 1  
  
    if ( b < 0 ):  
        return 0  
    for i in range(0, b):  
        returnValue *= a  
    return returnValue
```

```
|  
main()
```

Unlike other programming languages there is no special significance to the word main, however it is a good habit as it helps us organize and remain consistent



- A function is a reusable piece of code that is part of a larger program
- Executed when it is called by another line of code





How do functions help?

- Modularity – repeating code effectively
- Encapsulation – scoping/hiding – variables local - only exist within the block
- Abstraction – don't sweat the details

Let's look at our example again

mysteryFunctionScope.py - D:/SCAD_ALL/BuildWebSite/SRCWebSite/Pytho

File Edit Format Run Options Window Help

```
# Mystery Function example
```

```
badidea = 9
```

```
def main():
```

```
    num1 = 5
```

```
    num2 = 7
```

```
    # print a
```

```
    print badidea
```

```
    result = mysteryFunction(num1,num2)
```

```
    print "The number is " + str(result)
```

```
# Functions should also include a comment
```

```
def mysteryFunction(a,b):
```

```
    returnValue = 1
```

```
    print badidea
```

```
    # print num1
```

```
    if ( b < 0 ):
```

```
        return 0
```

```
    for i in range(0,b):
```

```
        returnValue *= a
```

```
    return returnValue
```

```
print badidea|
```

```
# print num1
```

```
main()
```

Let's look at our example again

The commented out lines all will produce an error

```
mysteryFunctionScope.py - D:/SCAD_ALL/BuildWebSite/SRCWebSite/Pytho
File Edit Format Run Options Window Help
# Mystery Function example

badidea = 9

def main():
    num1 = 5
    num2 = 7
    # print a
    print badidea

    result = mysteryFunction(num1,num2)
    print "The number is " + str(result)

# Functions should also include a comment
def mysteryFunction(a,b):
    returnValue = 1

    print badidea
    # print num1
    if ( b < 0 ):
        return 0
    for i in range(0,b):
        returnValue *= a
    return returnValue

print badidea
# print num1
main()
```

This is because
they are not
accessible

```
mysteryFunctionScope.py - D:/SCAD_ALL/BuildWebSite/SRCWebSite/Pytho
File Edit Format Run Options Window Help
# Mystery Function example

badidea = 9

def main():
    num1 = 5
    num2 = 7
    # print a
    print badidea

    result = mysteryFunction(num1,num2)
    print "The number is " + str(result)

# Functions should also include a comment
def mysteryFunction(a,b):
    returnValue = 1

    print badidea
    # print num1
    if ( b < 0 ):
        return 0
    for i in range(0,b):
        returnValue *= a
    return returnValue

print badidea
# print num1
main()
```

badidea is what is
termed a global
variable

It is accessible
everywhere

```
mysteryFunctionScope.py - D:/SCAD_ALL/BuildWebSite/SRCWebSite/Pytho
File Edit Format Run Options Window Help
# Mystery Function example

badidea = 9

def main():
    num1 = 5
    num2 = 7
    # print a
    print badidea

    result = mysteryFunction(num1,num2)
    print "The number is " + str(result)

# Functions should also include a comment
def mysteryFunction(a,b):
    returnValue = 1

    print badidea
    # print num1
    if ( b < 0 ):
        return 0
    for i in range(0,b):
        returnValue *= a
    return returnValue

print badidea
# print num1
main()
```



Unless absolutely necessary **global variables** are not good coding style



Homework:

Continue working on your quilting exercise
due Class 6