FUNCTIONS

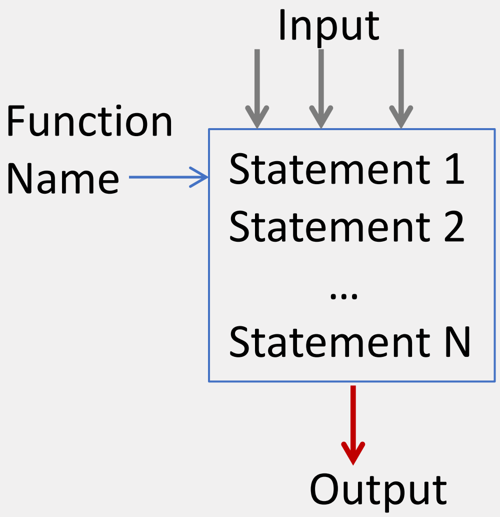
CIS 41A – INTRODUCTION TO PROGRAMMING IN PYTHON

DE ANZA COLLEGE BASED ON MATERIALS BY CLARE NGUYEN

What Is a Function? (1 of 2)

- A function is a group of Python statements that work together to perform a task.

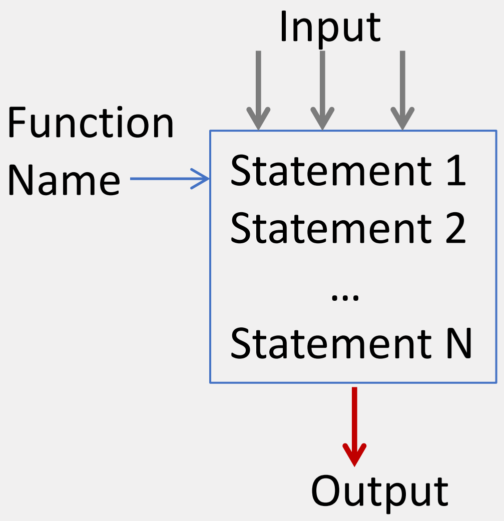
- Properties of a function:
  - Contains multiple statements
  - Has a name
  - Can accept one or more input data to work with them
  - Can produce one output data



What Is a Function? (2 of 2)

- A function is a group of Python statements that work together to perform a task.

- Examples :
  - A function that calculates the gas mileage of a car
  - A function that darkens the color of an image
  - A function that counts the number of words in a text file
  - A function that prints text to screen



Input

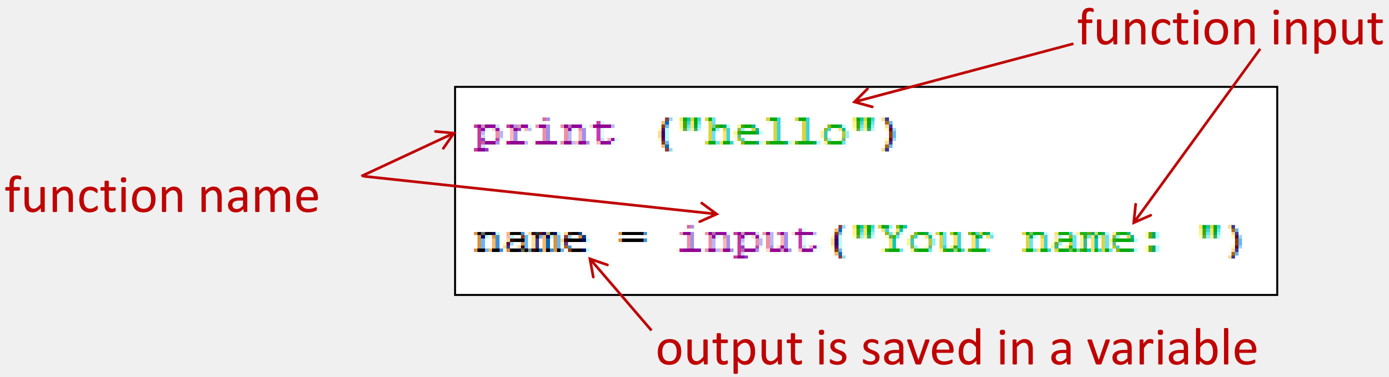
Function Name

Statement 1
Statement 2
…
Statement N

Output

Built-in Function

- Some of the functions that we have worked with are:



```
print ("hello")

name = input("Your name: ")
```

function input

function name

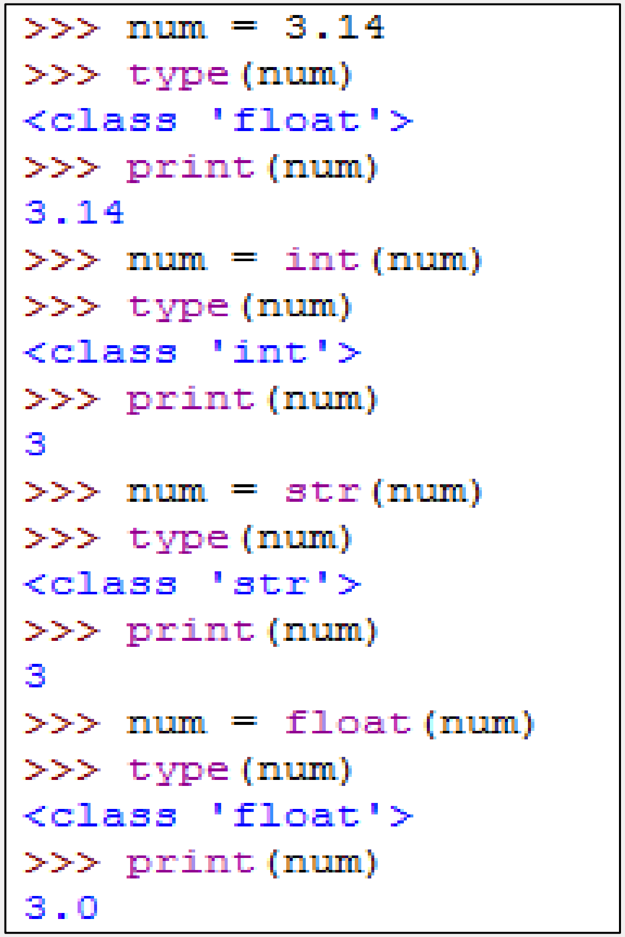
output is saved in a variable

- A function name followed by ( ) in a Python statement, we *call* the function and cause the function to run.

- When a function runs, the statements that make up the function are run by the CPU one by one.

Built-in Functions (1 of 2)

- Python has many built-in functions. The following are some commonly used ones.



```
>>> num = 3.14
>>> type(num)
<class 'float'>
>>> print(num)
3.14
>>> num = int(num)
>>> type(num)
<class 'int'>
>>> print(num)
3
>>> num = str(num)
>>> type(num)
<class 'str'>
>>> print(num)
3
>>> num = float(num)
>>> type(num)
<class 'float'>
>>> print(num)
3.0
```

The type function:

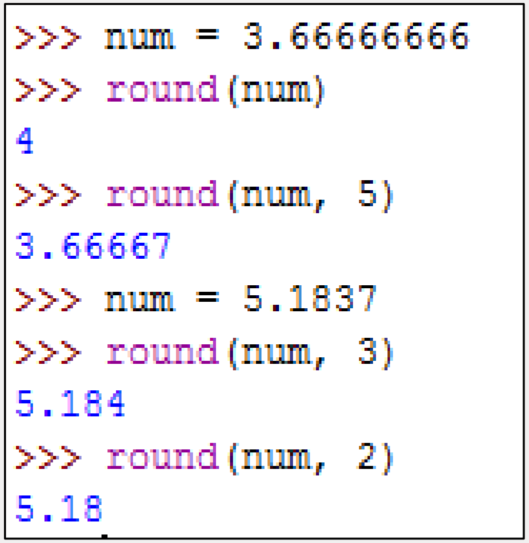
- accepts a data value as input
- returns (or outputs) the type of the data

The int, float, str conversion functions:

- accept a data value as input
- return the same data but as a new type

More Built-in Functions

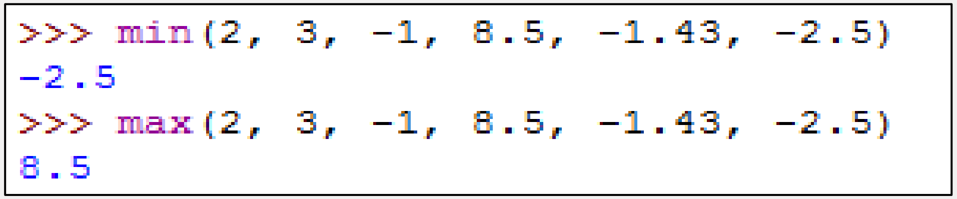
- round function



```
>>> num = 3.66666666
>>> round(num)
4
>>> round(num, 5)
3.66667
>>> num = 5.1837
>>> round(num, 3)
5.184
>>> round(num, 2)
5.18
```

(1) a float data value
(2 - optional) decimal places

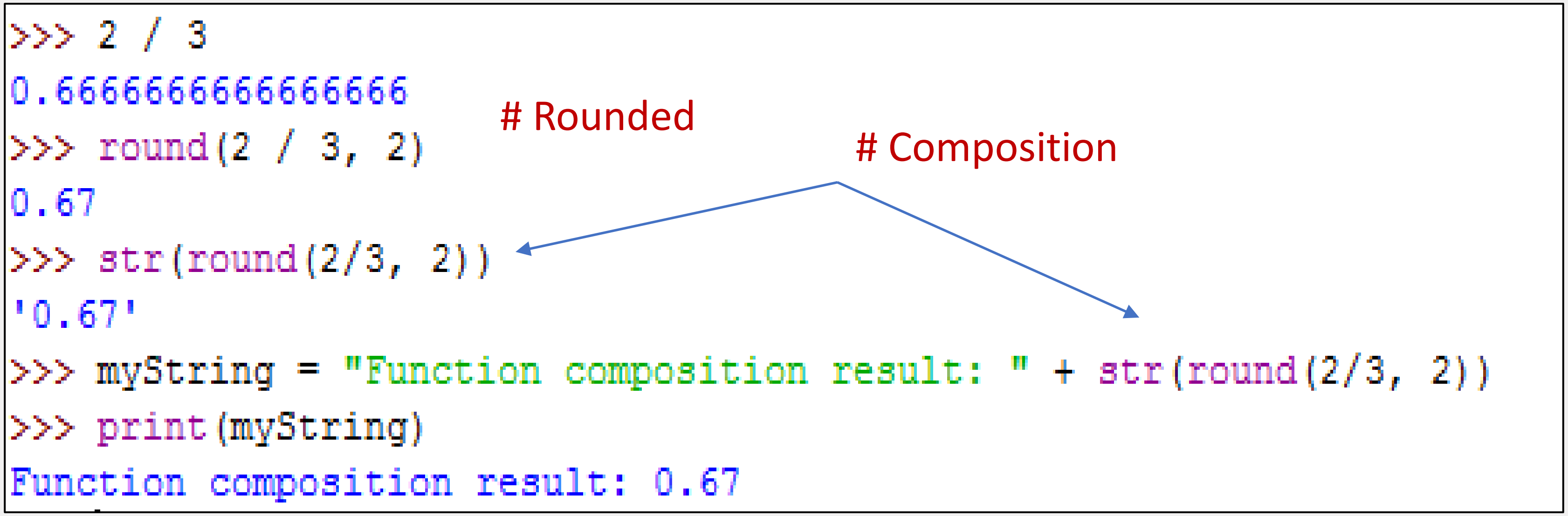
- min / max functions



```
>>> min(2, 3, -1, 8.5, -1.43, -2.5)
-2.5
>>> max(2, 3, -1, 8.5, -1.43, -2.5)
8.5
```

# Composition

- Python supports function chaining, which means it lets us chain function calls together.
- Example of using function composition to build a text string which is the rounded result of 2 / 3:



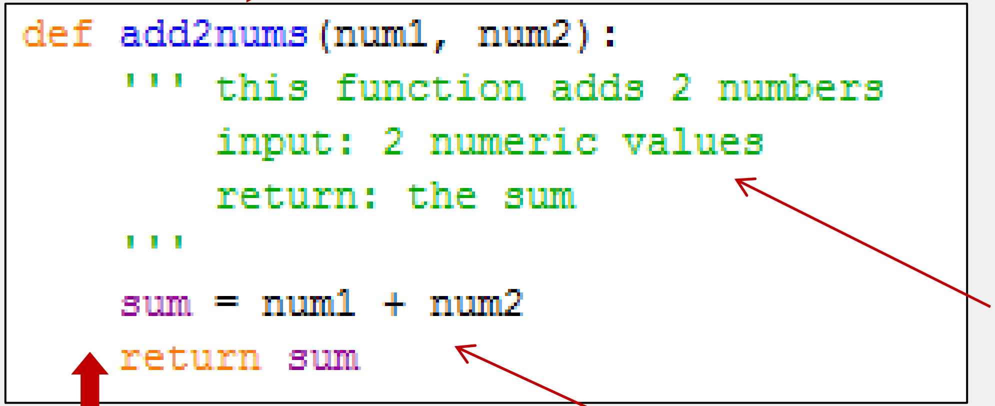
```
>>> 2 / 3
0.6666666666666666
>>> round(2 / 3, 2)
0.67
>>> str(round(2/3, 2))
'0.67'
>>> myString = "Function composition result: " + str(round(2/3, 2))
>>> print(myString)
Function composition result: 0.67
```

User-defined Functions

1. Start with def (for define)

2. Function name

3. Input parameters inside ( ), and end with :



```
def add2nums(num1, num2):
    ''' this function adds 2 numbers
        input: 2 numeric values
        return: the sum
    '''
    sum = num1 + num2
    return sum
```

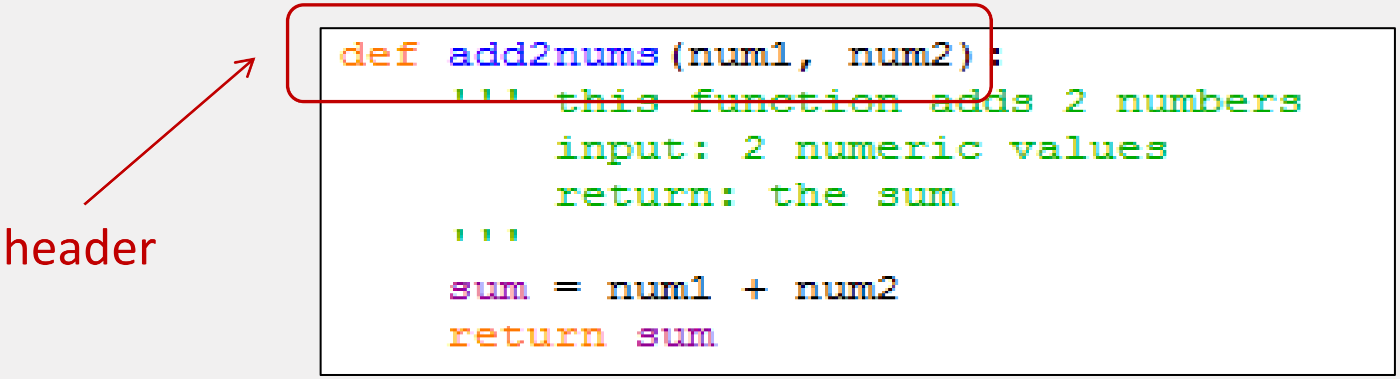
4. Add comments to describe the function

!! Important !!
Indent and line up all statements

5. Add statements that do the work

Function Definition (1 of 2)

- The *function definition* is the block of code that makes up the function.

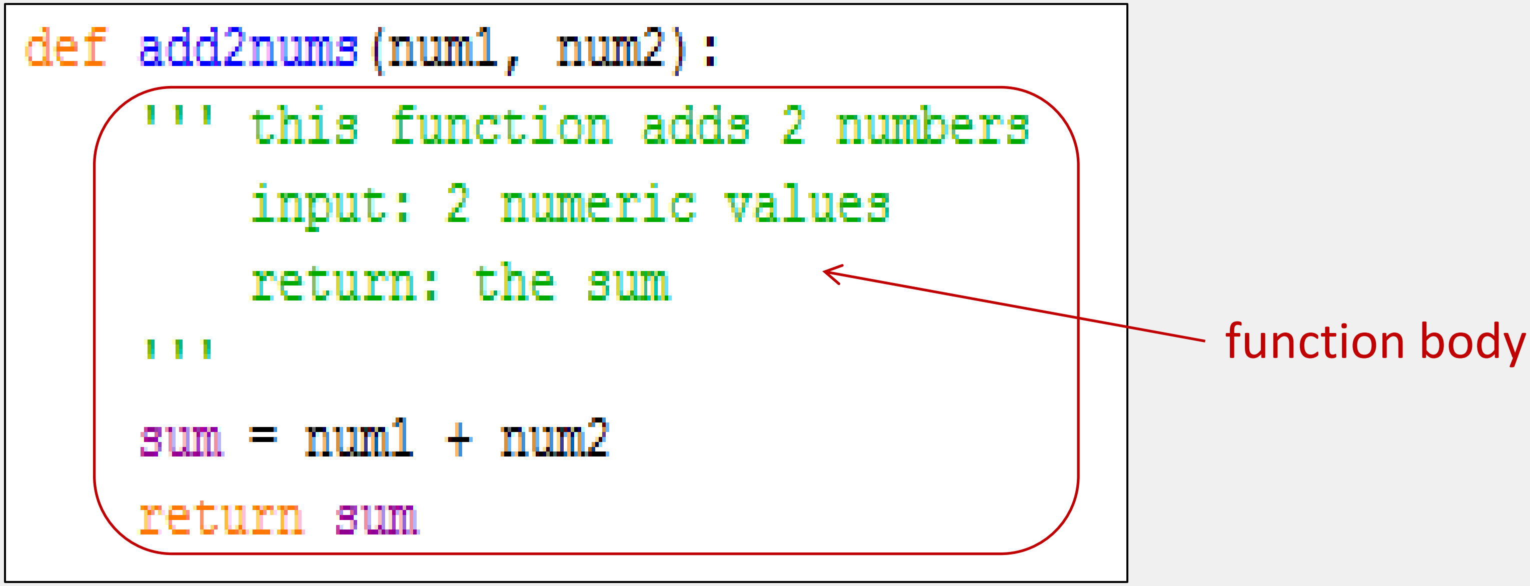


```
def add2nums(num1, num2):
    ''' this function adds 2 numbers
        input: 2 numeric values
        return: the sum
    '''
    sum = num1 + num2
    return sum
```

header

- A function definition starts with a *function header*, and has 3 parts:
    - The keyword def, define this block of statement with a name
    - A descriptive function name
    - A list of input *parameters*, separated by comma

Function Definition (2 of 2)



```
def add2nums(num1, num2):
    ''' this function adds 2 numbers
        input: 2 numeric values
        return: the sum
    '''
    sum = num1 + num2
    return sum
```

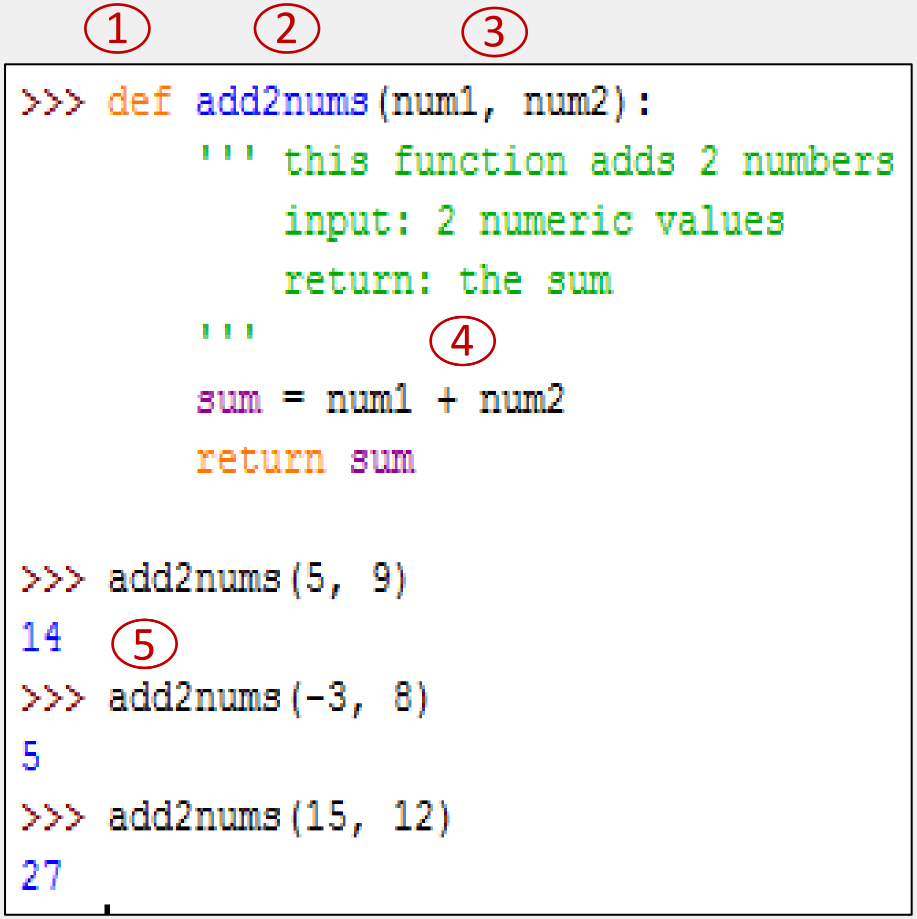
function body

Following the header is the *function body*:

- Indent the function body with the exact same spacing.
- List of Python statements that make up the function.
- If the function has an output, use the return keyword for the output value.

Flow of Execution

1. The function definition
2. Function name
3. Input *parameters* num1 and num2 of add2nums.
4. The code of add2nums producing a sum
5. Output

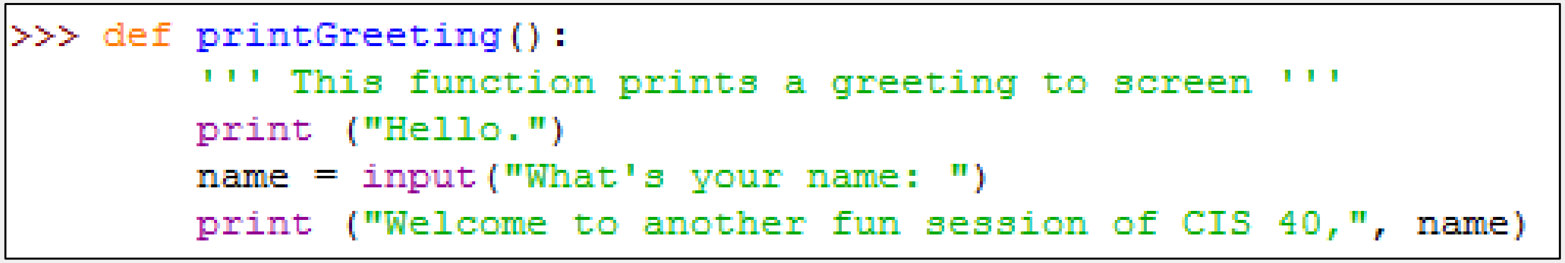


```
>>> def add2nums(num1, num2):
        ''' this function adds 2 numbers
            input: 2 numeric values
            return: the sum
        '''
        sum = num1 + num2
        return sum

>>> add2nums(5, 9)
14
>>> add2nums(-3, 8)
5
>>> add2nums(15, 12)
27
```

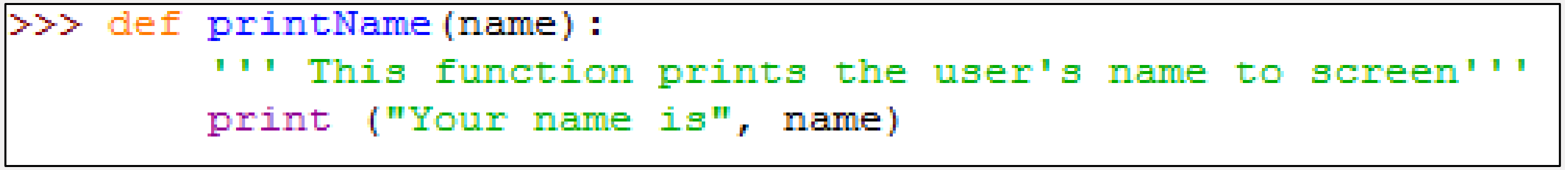
Function IO (1 of 2)

- The function add2nums has 2 input arguments and 1 return value.
- Example of a function that has no input argument and no return value:



```
>>> def printGreeting():
        ''' This function prints a greeting to screen '''
        print ("Hello.")
        name = input("What's your name:  ")
        print ("Welcome to another fun session of CIS 40,", name)
```

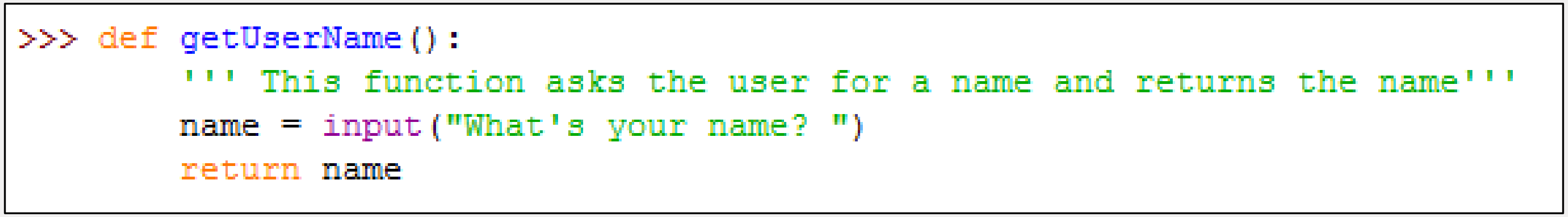
- Example of a function that has 1 input argument and no return value:



```
>>> def printName(name):
        ''' This function prints the user's name to screen'''
        print ("Your name is", name)
```

Function IO (2 of 2)

- Example of a function that has no input argument and 1 return value



```
>>> def getUserName():
        ''' This function asks the user for a name and returns the name'''
        name = input("What's your name? ")
        return name
```