Lecture 9
Functions – Part 2

Objectives

- After completing the lesson, the student will be able to:
  - Call user defined functions without values.
  - Call user defined functions with values.
  - Return values from a user defined function.
  - Describe the output of functions that pass arguments and return values.

Defining a User Defined Functions

- Rules to define a function in Python.
  - Function blocks begin with the keyword def followed by the function name and parentheses ( ).
  - Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
  - The first statement of a function can be the documentation string of the function or docstring.
  - The code block within every function starts with a colon (:) and is indented.
  - The statement return (expression) exits a function.

Syntax of the User Defined Function

```
def functionname( parameters ):
      #documentation of the function or docstring
      statements of the function
      return (expression)
```

Example: This function takes a string as input parameter and prints it on standard screen.

```
def printstring(name):
    # This function prints the passed string into this function
    print(name)
    return
printstring("Science Faculty")
```

Calling a Function without Arguments

- Example: Find area of a circle

```
def area():
    pi = 3.14
    radius = 2
    ar = pi*pow(radius,2)
    print("Area of the cirlce =", ar)


def circle():
    area()
```

Executing Functions

- When Python comes to a function call, it initiates a four-step process.
  - The calling program suspends at the point of the call.
  - The formal parameters of the function get assigned the values supplied by the actual parameters in the call.
  - The body of the function is executed.
  - Control returns to the point just after where the function was called.

Passing a Value (Parameter)

- A parameter is a variable that is initialized when the function is called.

- Example:

```
def cake(person):
    birthday()
    birthday()
    print("Happy Birthday, dear", person,".")
    birthday()
```

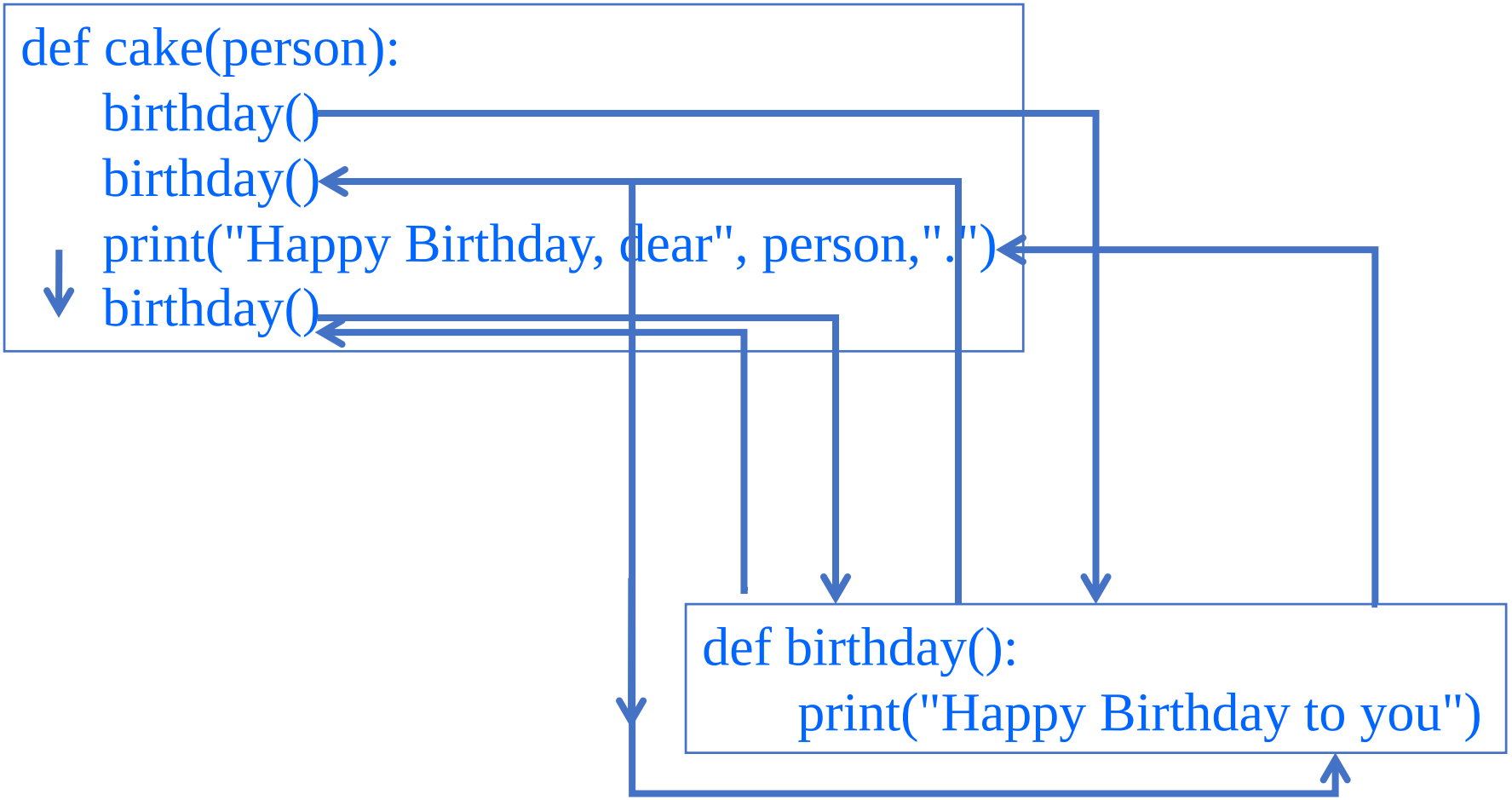
Passing a Value (Parameter)

- Example:

```
def birthday():
   print("Happy Birthday to you")
```

  - Type `cake("SSS")`, at the command prompt to pass SSS as the value of the person parameter.

Passing a Value (Parameter)



Calling Functions with values

```
def area(x,y):
   print("Area of the rectangle = ", (x*y))

def perimeter(x,y):
   print("Perimeter of the rectangle = ", (2*x+2*y))

def rectangle():
   x = int(input("Enter the length: "))
   y = int(input("Enter the breadth: "))
   area(x,y)
   perimeter(x,y)
```

Calling Functions with values

- The rectangle() function is used to call area() and perimeter() functions with the values of x and y.
- During the execution of the function, it asks to enter the values of x and y from the keyboard.
- These values are passed as arguments to individual functions.
- Finally, it calculates and display the result.

Return values from functions

- We can call a function many times and get different results by changing the input parameters.

- Values can be returned to the caller by using the return statement.

- Example: consider the call to the sqrt() function from the math library.

```
root = math.sqrt(b*b - 4*a*c)
```

  - sqrt returns the square root of its argument.

Return values from functions

- Example: Returns sum of the two numbers

```
def add():
      a = int(input("Enter the first value: "))
      b = int(input("Enter the second value: "))
      c = s(a,b)
      print("Sum of the two numbers = ", c)

def s(x,y):
      return(x+y)
```

Program Execution

- Type this at the command prompt and press enter.

```
add()
```

  - This will execute add function and asks the user to input two values from the keyboard.

```
Enter the first value: 2
Enter the second value: 3
Sum of the two numbers =  5
```

Return values from functions

- Example: Returns sum of the two numbers

```
def add(a,b):
    c = s(a,b)
    print("Sum of the two numbers = ", c)

def s(x,y):
    return(x+y)
```

Program Execution

- Type this at the command prompt and press enter.

```
add(3,4)
```

  - This will pass 3 and 4 as arguments to add function.
  - The value 3 will be assigned to variable a and the value 4 will be assigned to variable b.

```
Sum of the two numbers =  7
```

Return values from functions

- Function can return more than one value.
- This can be done by listing more than one expression in the return statement.

- Example
  - `def main():`
    - `x = int(input("Enter first number: "))`
    - `y = int(input("Enter second number: "))`
    - `s,d = SumDiff(x,y)`
    - `print("The sum is ", s, "and the difference is" , d)`

Return values from functions

```
def SumDiff(a,b):
    s = a+b
    diff = a-b
    return s, diff
```

- Type main() at the command prompt and press enter to run it.

- Enter two numbers separated by comma when it asks to enter numbers.

Default Arguments

- A default argument is an argument that assumes a default value if a value is not provided in the function call for that argument.

- Following example print default age if it is not passed.

```
def information( name = “Ali”, age = 35 ):
   #This prints a passed information into this function
   print("Name  = ", name)
   print("Age   = ", age)
   return

# Now you can call  information function
information( age=50, name="Hassan Ahmed" )
information( name="Hassan Ahmed" )
information( name="Hassan Ahmed" )
```

Scope of Variables

- All variables in a program may not be accessible at all locations in that program. This depends on where you have declared a variable.

- The scope of a variable determines the portion of the program where you can access a particular identifier.

- There are two basic scopes of variables in Python:
  - Global variables - variables that are defined outside a function body.
  - Local variables - variables that are defined inside the function body.

- This means that local variables can be accessed only inside the function in which they are declared, whereas global variables can be accessed throughout the program body by all functions.

Local Variables and Global Variables

```
total = 0
def add( arg1, arg2):
    #Add both the parameters and return them.
    total = arg1 + arg2;
    print("Inside the function local total : ", total)
    return total;
```

- # Now you can call the 'add' function

```
add( 10, 20 );
print("Outside the function global total : ", total)
```