Fruitful function

Presented By
M.Malarmathi
AP/IT

- Function that returns value is called fruitful functions
- The first example is area, which returns the area of a circle with the given radius:

```
def area(radius):
temp = math.pi * radius**2
return temp
```

- return statement includes an expression.
- This statement means: “Return immediately from this function and use the following expression as a return value.”

- Function can written more concisely:

```
def area(radius):
return math.pi * radius**2
```

- multiple return statements, one in each branch of a conditional:

```
def absolute_value(x):
if x < 0:
return -x
else:
return x
```

return statements are in an alternative conditional, only one will be executed

Examples

```
def distance(x1, y1, x2, y2):
dx = x2 - x1
dy = y2 - y1
dsquared = dx**2 + dy**2
result = math.sqrt(dsquared)
return result
```

Local and Global scope

- 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
  - Local variables

TYPES

When it is accessible from all parts of a program (ie. Anywhere!) Variables defined outside a function body **GLOBAL Scope**	When it is accessible **ONLY** within a function/procedure(: Variables that are defined inside a function body) **LOCAL Scope**

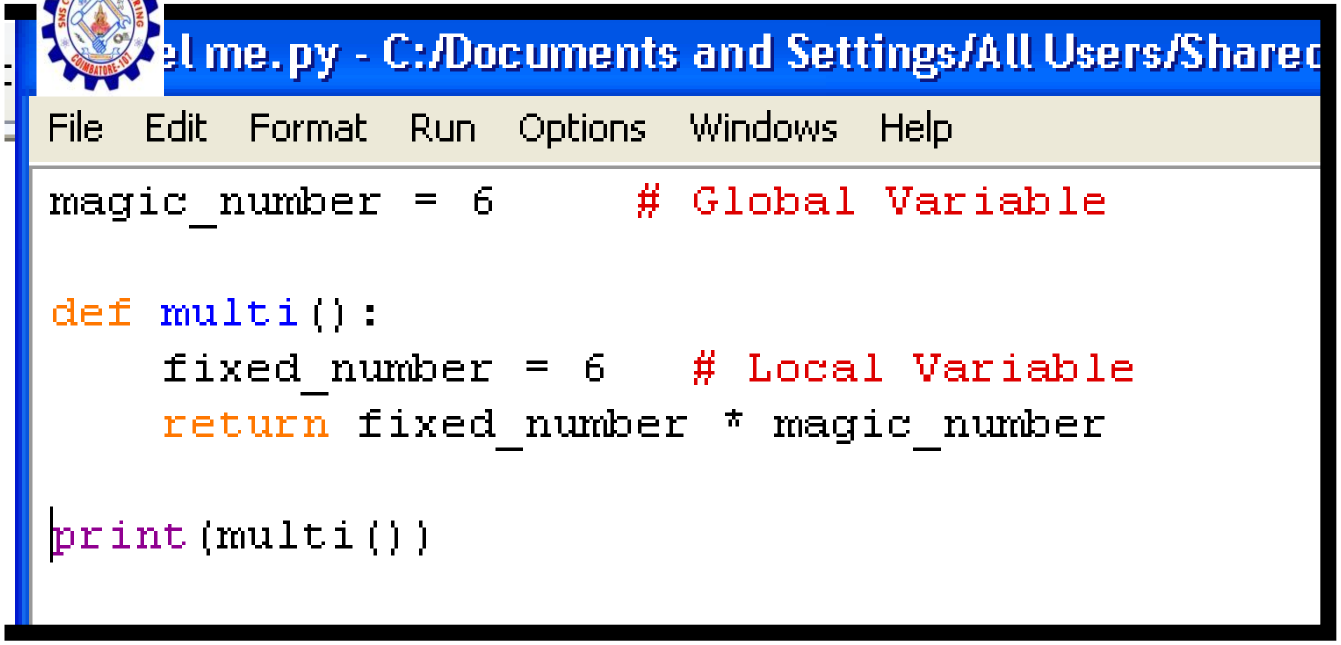
```
• name = ‘Mary Poppins’

• print(name)
```

```
def Greeting():
    name = 'Mary
Poppins'
    age = 200
    address = 'Far Far
away'
```

declared ***OUTSIDE*** of any ***functions-----Global***

declared inside a function-----Local



el me.py - C:/Documents and Settings/All Users/Sharec

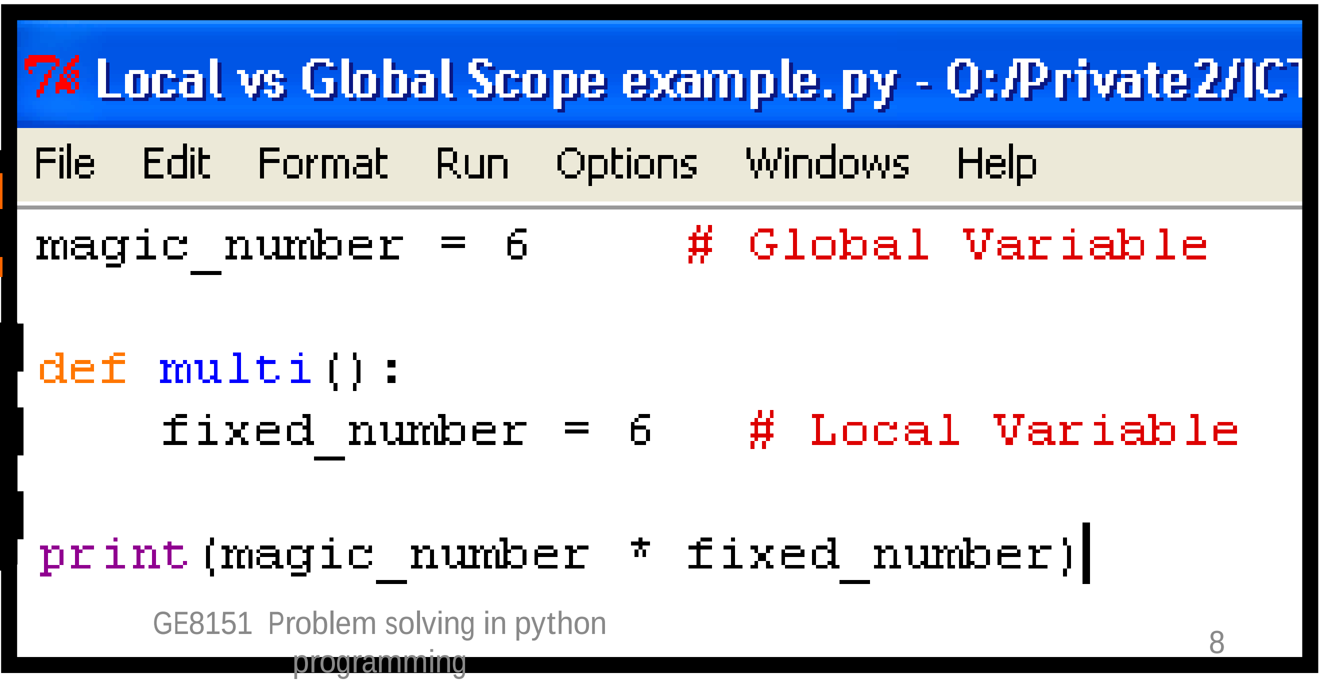
```
magic_number = 6       # Global Variable

def multi():
    fixed_number = 6   # Local Variable
    return fixed_number * magic_number

print(multi())
```

Result ?

36



Local vs Global Scope example.py - O:/Private2/ICT

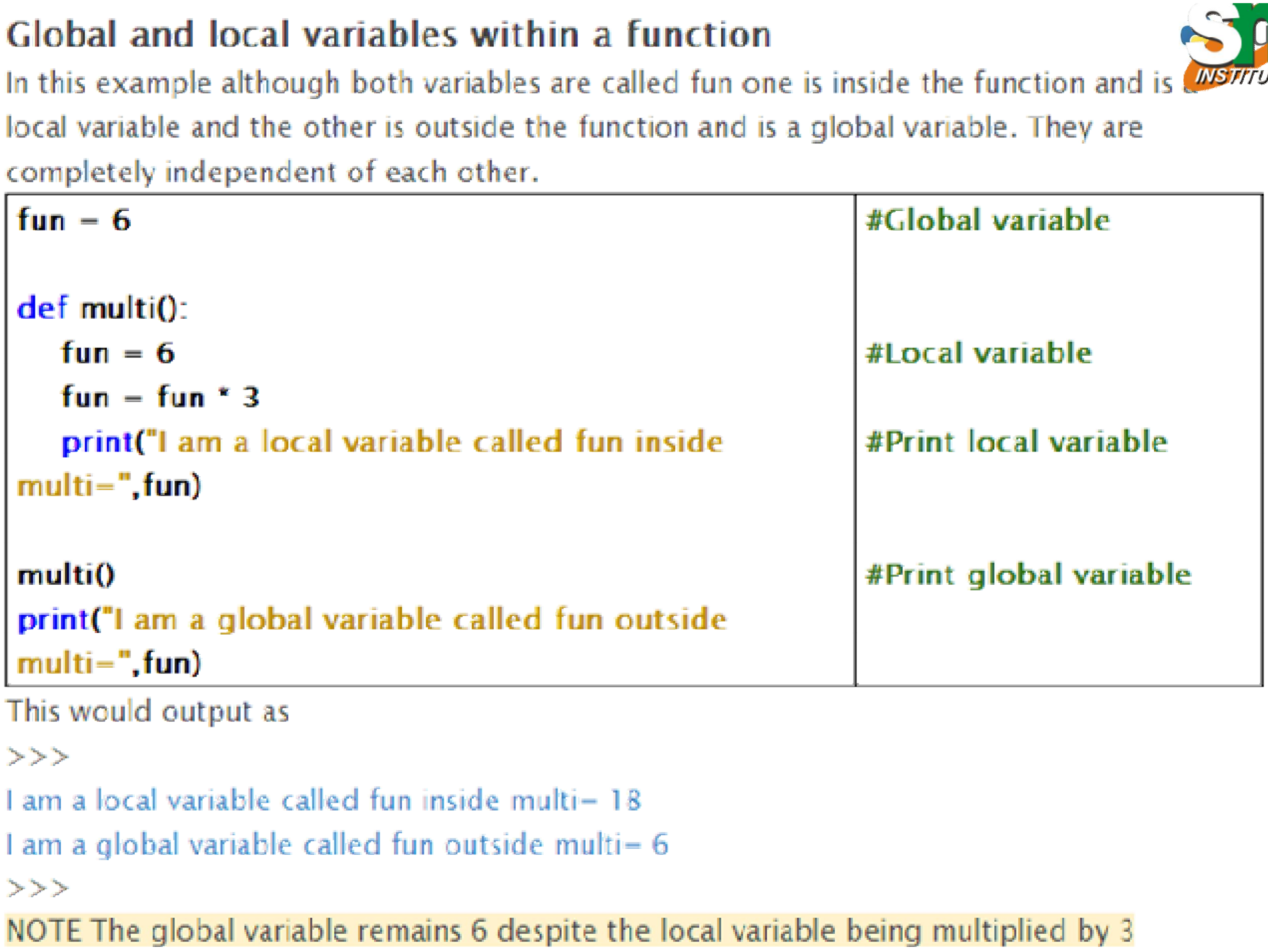
```
magic_number = 6       # Global Variable

def multi():
    fixed_number = 6   # Local Variable

print(magic_number * fixed_number)
```

Result ?

```
NameError: name
'fixed_number' is
not defined
```



Global and local variables within a function

In this example although both variables are called fun one is inside the function and is a local variable and the other is outside the function and is a global variable. They are completely independent of each other.

```
fun = 6                                                  #Global variable

def multi():
    fun = 6                                              #Local variable
    fun = fun * 3
    print("I am a local variable called fun inside       #Print local variable
multi=",fun)

multi()                                                  #Print global variable
print("I am a global variable called fun outside
multi=",fun)
```

This would output as

```
>>>
I am a local variable called fun inside multi= 18
I am a global variable called fun outside multi= 6
>>>
```

NOTE The global variable remains 6 despite the local variable being multiplied by 3

Composition

- Call one function from within another. This ability is called **composition.**
- The center point is stored in the variables xc and yc, and the perimeter point is in xp and yp. The first step is to find the radius of the circle, which is the distance between the two points. We just wrote a function, distance, that does that:

```
radius = distance(xc, yc, xp, yp)
```

- The next step is to find the area of a circle with that radius; we just wrote that, too:

```
result = area(radius)
```

- Encapsulating these steps in a function:

```
def circle_area(xc, yc, xp, yp):
radius = distance(xc, yc, xp, yp)
result = area(radius)
return result
```

- The temporary variables radius and result are useful for development and debugging, but once the program is working, we can make it more concise by composing the function calls:

```
def circle_area(xc, yc, xp, yp):
return area(distance(xc, yc, xp, yp))
```

Thank You