COMP7035

Python for Data Analytics and Artificial Intelligence

Functions

Renjie Wan, Jun Qi

17/09/2024

香港浸會大學
HONG KONG BAPTIST UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE
計算機科學系

Simple example

Example: Suppose we want to find the circumference of a circle with radius 2.5. We could write

```
import math
radius = 2.5
circumference = 2*math.pi * radius
```

A function to calculate the circumference:

$$A(\boldsymbol{r}) = 2\pi\boldsymbol{r}$$

Functions: From Math to Python

- Functions are used to abstract components of a program.
- Much like a mathematical function, they just take some input and then do something to find the result.

$$f(x) = x^2 + 1$$

A function in math:
f : function name
x is the input of the function.
x is manipulated by the operation on the righthand side of "="

Can we transfer this from math to Python? What do we need to change?

x^2 can only be represented as x**2 in Python.

$x^2 + 1$ ⟹ `x**2 + 1`

Functions: From Math to Python

- Functions are used to abstract components of a program.
- Much like a mathematical function, they just take some input and then do something to find the result.

$$f(x) = x^2 + 1$$

A function in math:
f : function name
x is the input of the function.
x is manipulated by the operation on the righthand side of "="

Can we transfer this from math to Python? What do we need to change?

x^2 can only be represented as x**2 in Python.

```
f(x): x**2 + 1
```
⊗

```
def f(x): x**2 + 1
```
OK NOW

A function in Python:
f : function name
x is the input of the function.
x is manipulated by the operation inside the function

We need to use def to tell Python that this is a function

Functions: From Math to Python

- Functions are used to abstract components of a program.
- Much like a mathematical function, they just take some input and then do something to find the result.

$$f(x) = x^2 + 1$$

A function in math:
f : function name
x is the input of the function.
x is manipulated by the operation on the righthand side of "="

Can we transfer this from math to Python? What do we need to change?

x^2 can only be represented as x**2 in Python.

```
def f(x):
    result = x**2 + 1
    print(result)
```

A function in Python:
f : function name
x is the input of the function.
x is manipulated by the operation inside the function

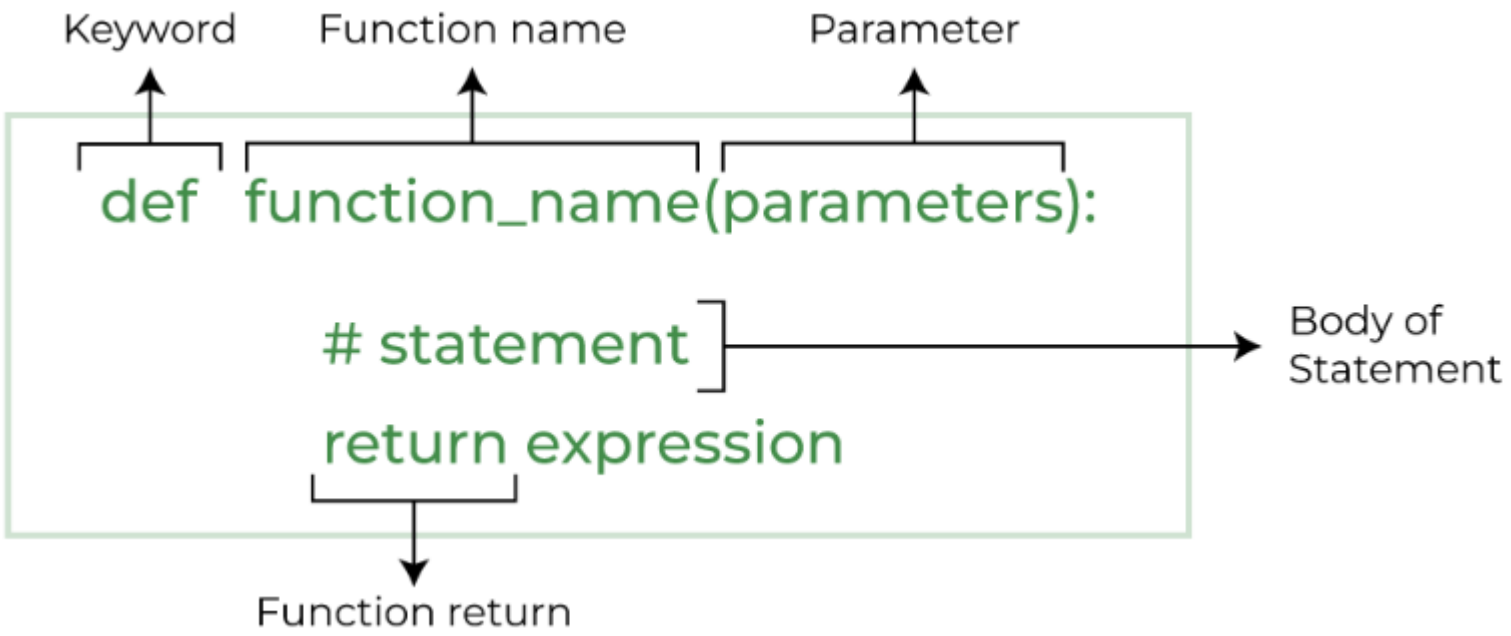
We need to use def to tell Python that this is a function

Functions: def

- Define your own functions using the keyword **def**
- Then comes the function name, with arguments in braces, and then a colon
- Parameters (arguments) through which we pass values to a function.
- A colon to mark the end of the function header
- Optional documentation string to describe what the function does
- One or more python statements that make up the function body. Statements must have the same indentation level.

```
def func(arg1, arg2):
    """docstring"""
    statements(s)
```

Functions in Python



Function can be more in python

```
# function definition
def greet(name):
    """
    This function greets to
    the person passed in as
    a parameter
    """
    print("Hello, " + name + ". Good morning!")
```

How to call a function?

- Once we have defined a function, we can call it from another function, program, or even the Python prompt.
- To call a function we simply type the function name with appropriate parameters.

```
# function definition
def greet(name):
    """
    This function greets to
    the person passed in as
    a parameter
    """
    print("Hello, " + name + ". Good morning!")

# function call
greet('Paul')
```

Result: Hello, Paul. Good morning!

How to call a function?

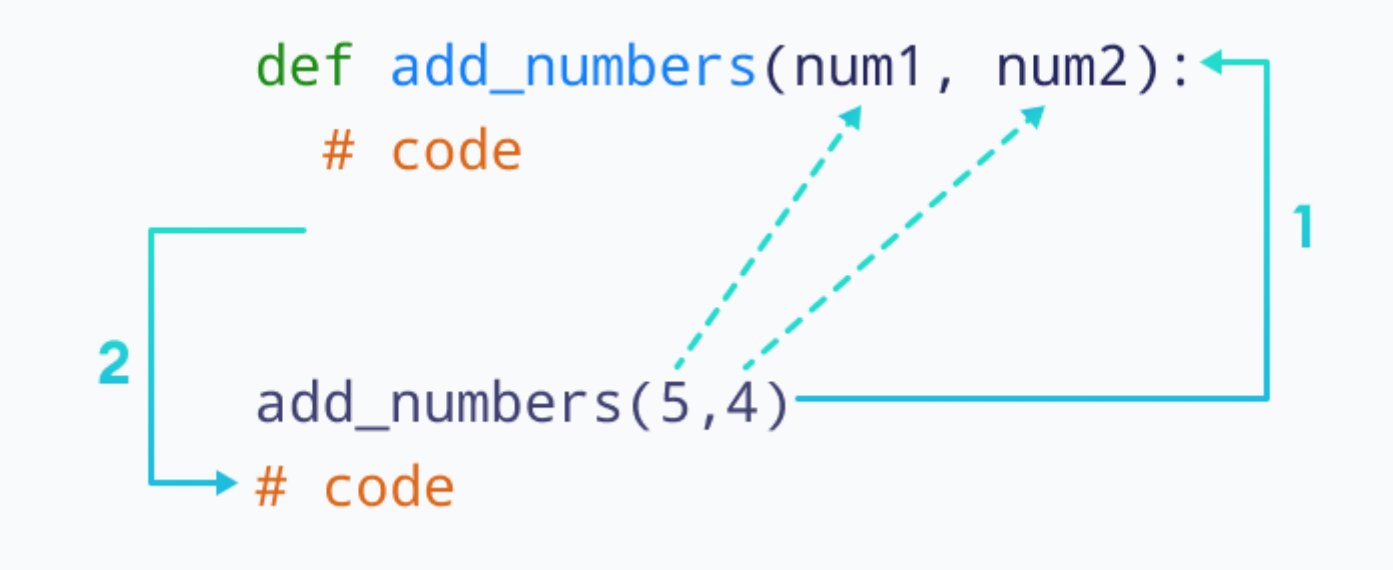
- You must define the function before you can call it. Otherwise, you will encounter errors.

```
# function call
greet2('Paul')

# function definitior
def greet2(name):
    """
    This function greets to
    the person passed in as
    a parameter
    """
    print("Hello, " + name + ". Good morning!")
```

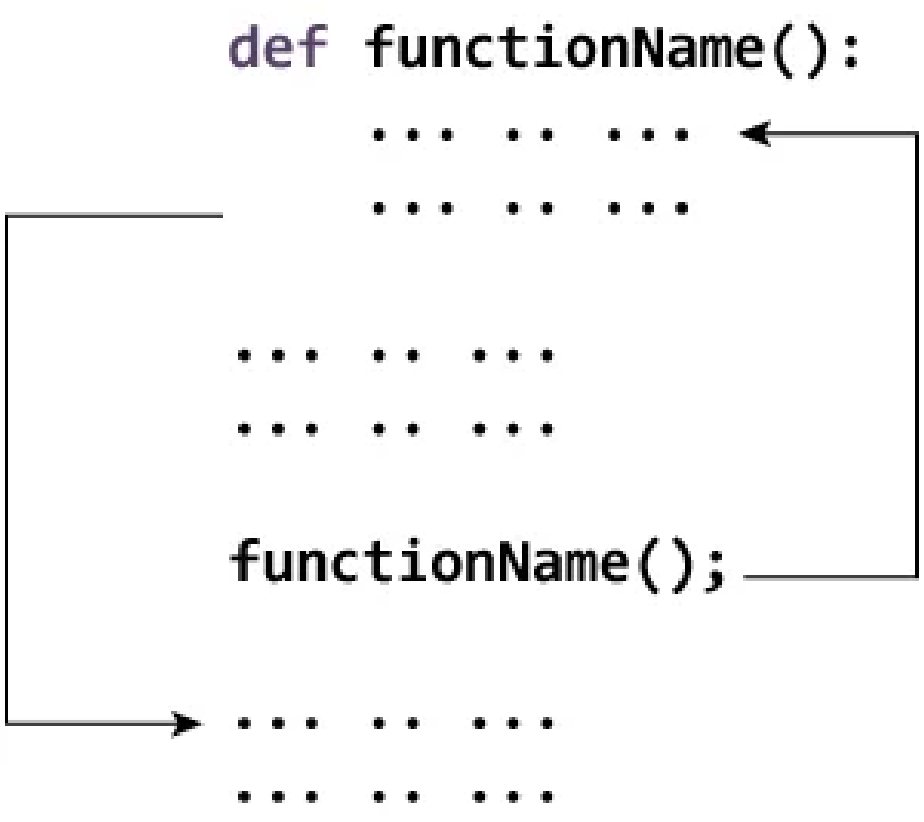
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
<ipython-input-4-4c4bc1ea0c0a> in <module>
      1 # function call
----> 2 greet2('Paul')
      3
      4 # function definition
      5 def greet2(name):

NameError: name 'greet2' is not defined
```

How to call a function?

```
# function with two arguments
def add_numbers(num1, num2):
    sum = num1 + num2
    print("Sum: ",sum)

# function call with two values
add_numbers(5, 4)

# Output: Sum: 9
```



How does it works in Python?




```
def judgeNum(num):
    if num > 0:
        print(num)
    else:
        print(num**2)
judgeNum(1)
judgeNum(-2)
judgeNum(-3)
```

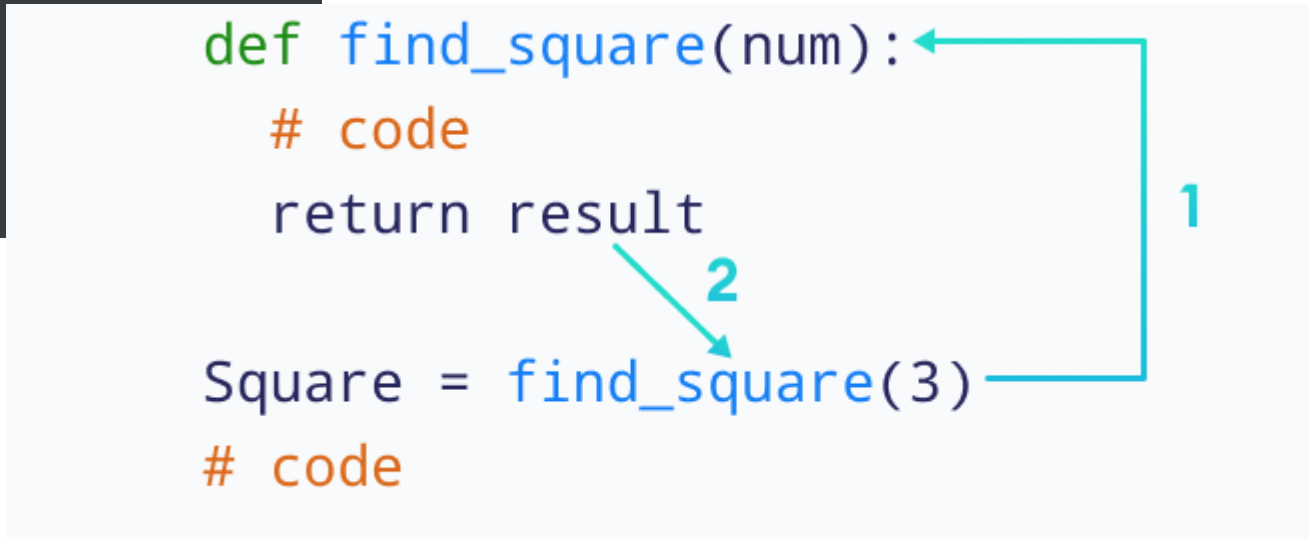
Functions: return

- "return" statement is a special statement that you can use inside a function or method to send the function's result back to the caller.
- A "return" statement consists of the return keyword followed by an optional return value.
- Once Python hits return, it will return the output and jump out of the function

```
def calc_circumference(radius):
    circumference = 2*math.pi * radius
    return circumference

result = calc_circumference(5)
```

Functions: return

```
# function definition
def find_square(num):
    result = num * num
    return result

# function call
square = find_square(3)

print('Square:',square)

# Output: Square: 9
```



Functions: return

- You can also return several values at the same time
  - You can also input several parameters into one function

```
def simple(a,b):
    return a+b, a*b

result1, result2 = simple(1,2)
def simple(a):
    return a,a+1,a+2,a+3

result1,result2,result3,result4 = simple(1)
```

Why do we need Function?

- Make your code simple

```
num = 1
if num>0:
  print(num)
else:
  print(num**2)
num2 = -2
if num2>0:
  print(num2)
else:
  print(num2**2)
num3 = -3
if num3>0:
  print(num2)
else:
  print(num3**2)
```

```
def judgeNum(num):
  if num > 0:
    print(num)
  else:
    print(num**2)
judgeNum(1)
judgeNum(-2)
judgeNum(-3)
```

Why do we need Function?

```
def isYourCorrect(whatyousay):
    if (whatyousay =='Y' or whatyousay =='y'):
        print("YOU SAID YES")
    elif(whatyousay =='N' or whatyousay =='n'):
        print("YOU SAID NO")
    else:
        print("INVALID INPUT")
isYourCorrect('Y')
isYourCorrect('N')
isYourCorrect('dadasda')
```

elif: shortname of elseif. If the condition for if is False , it checks the condition of the next elif block and so on. If all the conditions are False , the body of else is executed.

Try Some Experiments

Python Lambda

- A lambda function is a small anonymous function.

```
x = lambda a : a + 10
print(x(5))
```

Add 10 to argument a, and return the result

```
x = lambda a, b : a * b
print(x(5, 6))
```

Multiply argument a with argument b and return the result

Define values in the header

- You can define the parameter value in the function header
  - Once it is defined in header, it becomes the default value.

```
def g(a, x, b=0):
    return a * x + b
g(1,1) #equivalent to g(1,1,0)
def h(a, b, x=3,y=3):
    return a * x + b*y

h(1,1) #equivalent to h(1,1,3,3)
```

Scope and Lifetime of variables

- Scope of a variable is the portion of a program where the variable is recognized.
- Parameters and variables defined inside a function are not visible from outside the function. Hence, they have a local scope.

```
def my_func():
    x = 10
    print("Value inside function:",x)


x = 20
my_func()
print("Value outside function:",x)
```

`x` here is a local scope, and it does not have any relationship with `x` outside

`x` here is defined outside function. It cannot be influenced by any operations inside function

Scope and Lifetime of variables

- Scope of a variable is the portion of a program where the variable is recognized.
- Parameters and variables defined inside a function are not visible from outside the function. Hence, they have a local scope.

```
x = 1
def add_one(x):
  x = x + 1     # local x
  return x
y = add_one(x) # x = 1, y = 2
```

Scope and Lifetime of variables

- When we define a variable inside the main program but outside the function body, then it has a global scope. The variable declared in the main body of the program or file is a global variable.
- In the global scope, the global variable will be visible throughout the program or file, and also inside any file which imports that file. We can easily access a variable defined in global scope from all kinds of functions and blocks.

Output:

```
Value of x from local scope =  50
Value of x from global scope =  50
```

```
# Declaring a global variable in the global scope.
x = 50
# Declare a simple function that prints the value of x.
def showMe():
    print('Value of x from local scope = ',x)#calling variable x inside the function.

# Main program.
showMe() # calling function.
print('Value of x from global scope = ',x)
```

Scope and Lifetime of variables

- When we define a variable inside the main program but outside the function body, then it has a global scope. The variable declared in the main body of the program or file is a global variable.
- In the global scope, the global variable will be visible throughout the program or file, and also inside any file which imports that file. We can easily access a variable defined in global scope from all kinds of functions and blocks.

```
# Declaring a global variable in the global scope.
x = 50
# Declare a simple function that prints the value of x.
def showMe():
    print('Value of x from local scope = ',x)#calling variable x inside the function.

# Main program.
showMe() # calling function.
print('Value of x from global scope = ',x)
```

Output:

Value of x from local scope = 50
Value of x from global scope = 50

Specify the argument name

- You can specify the argument name with values so that caller does not need to remember the order of parameters.

```
def student(firstname, lastname):
    print(firstname, lastname)

# Keyword arguments
student(firstname='Renjie', lastname='Wan')
student(lastname= 'Cheung', firstname='Bob')

student('Cheung', 'Bob')
```

How to control your function?

- Give your function more constraints:
  - We can use if or only other control measures.

```
def triangle_area(base, height):
  if base<0 or height < 0:
    print("Base and height must be non-negative")
    break
  return 0.5*base*height
triangle_area(-1,2)
```

Wrong, since break can only be used to break the look, this is not a loop

break is used to end a loop prematurely while return is the keyword used to pass back a return value to the caller of the function. If return is used without an argument it simply ends the function and returns to where the code was executing previously.

```
def triangle_area(base, height):
  if base<0 or height < 0:
    print("Base and height must be non-negative")
    return
  return 0.5*base*height
triangle_area(-1,2)
```

Correct, since break can only be used to break the look, we use return to leave the function