Lecture 14

Topics: *Chapter 6. Defining Functions*

6.1 The Function of Functions
6.2 Functions, Informally
6.3 Future Value with a Function
6.4 Functions and Parameters: The Exciting Details
6.5 Functions That Return Values

6.1 The Function of Functions

The programs we've written so far had just one function: `main`

We also used some pre-written (built-in) functions and methods:
`abs`, `len`, `eval`,
functions/ methods from the Python standard libraries:
`math.sqrt`, `string.split`, and
so forth.

Chapter 6 covers whys and hows of designing our own functions.

6.1 The Function of Functions

Why do we need functions?

- mainly to make our programs easier to write, read, test, and to update/modify.

We can think of a function as a *subprogram* – a small program inside of a program.

- we write a sequence of statements and give that sequence a ***name***;
- then these instructions can be executed at any point in the program by referring to the *function name*.

Functions

Function definition

```
def <name>(<formal parameters>):
    <body>
```

`name` – is an identifier

`formal parameters` – list (possibly empty) of variable names

`body` – is the body of the function, all formal parameters are accessible only in this part.

Functions

Function definition

```
def <name>(<formal parameters>):
    <body>
```

`name` – is an identifier

`formal parameters` – list (possibly empty) of variable names

`body` – is the body of the function, all formal parameters are accessible only in this part.

Example:

```
def add(a,b):
    return a+b
```

Functions

Function invocation

Then let's see how can we **invoke a function** (**function call**):
`<name>(<actual parameters>)`

`actual parameters` – are the the ones whose values are assigned to formal parameters

Functions

Function invocation

Then let's see how can we **invoke a function** (**function call**):
`<name>(<actual parameters>)`

`actual parameters` – are the the ones whose values are assigned to formal parameters

Example:

```
def add(a,b):
    return a+b

add(2,3)
```

function call
function invocation

5 is returned by function `add`

Functions

Example:

Let's write a program that takes three values from the user (decimal values) and returns their product, their sum, their absolute values, and their average.

Let's split our program into four *subprograms* (*functions*):

1st function – finds the product, `product`
2nd function – finds the sum, `sum_`
3rd function – finds their absolute values, and `absolute_values`
4th function – finds their average `average`

6.4 Functions and Parameters: The Exciting Details

Example:

Let's write a program that takes three values from the user (decimal values) and returns their product, their sum, their absolute values, and their average.

Questions:

How do we supply the data values to the functions?

through parameters

How many of those three values received from the user should be given to each function?

all three

6.4 Functions and Parameters: The Exciting Details

Example:

Let's write a program that takes three values from the user (decimal values) and returns their product, their sum, their absolute values, and their average.

A draft of the program:

```
def main():
   get three values from the user (a,b,c)

   call product(a,b,c)
   call sum_(a,b,c)
   call absolute_values(a,b,c)
   call average(a,b,c)

   say good bye to the user
```

6.4 Functions and Parameters: The Exciting Details

Example:

Let's write a program that takes three values from the user (decimal values) and returns their product, their sum, their absolute values, and their average.

A draft of the program:

```
def main():
   get three values from the user (a,b,c)

   call product(a,b,c)
   call sum_(a,b,c)
   call absolute_values(a,b,c)
   call average(a,b,c)

   say good bye to the user
```

see program four_numbers.py

6.4 Functions and Parameters: The Exciting Details

Control of flow:

When Python comes to a function call, it initiates four-steps process:

1. The calling program suspends execution at the point of the call;

2. The formal parameters of the function get assigned the values supplied by the actual parameters in the call;

3. The body of the function is executed;

4. Control returns to the point just after where the function was called.

6.4 Functions and Parameters: The Exciting Details

Control of flow:

```
def main():

    a = float(input("Please input a decimal number:"))
    b = float(input("Please input another decimal number:"))
    c = float(input("Please input the last decimal number:"))

    product(a,b,c)

    sum_(a,b,c)

...
```

6.4 Functions and Parameters: The Exciting Details

Control of flow:

```
def main():

    a = float(input("Please input a decimal number:"))
    b = float(input("Please input another decimal number:"))
    c = float(input("Please input the last decimal number:"))

    product(a,b,c)

    sum_(a,b,c)

...
```

```
Please input a decimal number:
```

6.4 Functions and Parameters: The Exciting Details

Control of flow:

```
def main():

    a = float(input("Please input a decimal number:"))
    b = float(input("Please input another decimal number:"))
    c = float(input("Please input the last decimal number:"))

    product(a,b,c)

    sum_(a,b,c)

...
```

```
Please input a decimal number:1.2
```

6.4 Functions and Parameters: The Exciting Details

Control of flow:

```
def main():

    a = float(input("Please input a decimal number:"))
    b = float(input("Please input another decimal number:"))
    c = float(input("Please input the last decimal number:"))

    product(a,b,c)

    sum_(a,b,c)

...
```

```
Please input a decimal number:1.2
Please input another decimal number:
```

6.4 Functions and Parameters: The Exciting Details

Control of flow:

```
def main():

    a = float(input("Please input a decimal number:"))
    b = float(input("Please input another decimal number:"))
    c = float(input("Please input the last decimal number:"))

    product(a,b,c)

    sum_(a,b,c)

...
```

```
Please input a decimal number:1.2
Please input another decimal number:-3.2
```

6.4 Functions and Parameters: The Exciting Details

Control of flow:

```
def main():

    a = float(input("Please input a decimal number:"))
    b = float(input("Please input another decimal number:"))
    c = float(input("Please input the last decimal number:"))

    product(a,b,c)

    sum_(a,b,c)

...
```

```
Please input a decimal number:1.2
Please input another decimal number:-3.2
Please input the last decimal number:
```

6.4 Functions and Parameters: The Exciting Details

Control of flow:

```
def main():

    a = float(input("Please input a decimal number:"))
    b = float(input("Please input another decimal number:"))
    c = float(input("Please input the last decimal number:"))

    product(a,b,c)

    sum_(a,b,c)

...
```

```
Please input a decimal number:1.2
Please input another decimal number:-3.2
Please input the last decimal number:-4.5
```

6.4 Functions and Parameters: The Exciting Details

Control of flow:

```
def main():

    a = float(input("Please input a decimal number:"))
    b = float(input("Please input another decimal number:"))
    c = float(input("Please input the last decimal number:"))

    product(a,b,c)

    sum_(a,b,c)

...
```

The function main suspends its execution

6.4 Functions and Parameters: The Exciting Details

Control of flow:

```
def main():

    a = float(input("Please input a decimal number:"))
    b = float(input("Please input another decimal number:"))
    c = float(input("Please input the last decimal number:"))

    product(a,b,c)

    sum_(a,b,c)

...
```

```
def product(x1,x2,x3):
    print("The product of three numbers {0} x
{1} x {2} = {3}".format(x1,x2,x3,x1*x2*x3))
```

The function main suspends its execution

x1 = a = 1.2

x2 = b = - 3.2

x3 = c = -4.5

6.4 Functions and Parameters: The Exciting Details

Control of flow:

```
def main():

    a = float(input("Please input a decimal number:"))
    b = float(input("Please input another decimal number:"))
    c = float(input("Please input the last decimal number:"))

    product(a,b,c)

    sum_(a,b,c)

...
```

```
def product(x1,x2,x3):
    print("The product of three numbers {0} x {1} x {2} = {3}".format(x1,x2,x3,x1*x2*x3))
```

```
The product of three numbers 1.2 x -3.2 x -4.5 = 17.28
```

6.4 Functions and Parameters: The Exciting Details

Control of flow:

```
def main():

    a = float(input("Please input a decimal number:"))
    b = float(input("Please input another decimal number:"))
    c = float(input("Please input the last decimal number:"))

→   product(a,b,c)

    sum_(a,b,c)

...
```

```
Please input a decimal number:1.2
Please input another decimal number:-3.2
Please input the last decimal number:-4.5
The product of three numbers 1.2 x -3.2 x -4.5 = 17.28
```

6.4 Functions and Parameters: The Exciting Details

Control of flow:

```
def main():

    a = float(input("Please input a decimal number:"))
    b = float(input("Please input another decimal number:"))
    c = float(input("Please input the last decimal number:"))

    product(a,b,c)

→   sum_(a,b,c)

...
```

The function main suspends its execution

6.4 Functions and Parameters: The Exciting Details

Control of flow:

```
def main():

    a = float(input("Please input a decimal number:"))
    b = float(input("Please input another decimal number:"))
    c = float(input("Please input the last decimal number:"))

    product(a,b,c)

    sum_(a,b,c)

...
```

```
def sum_(s1,s2,s3):
    print("The sum of three numbers {0} +
{1} + {2} = {3}".format(s1,s2,s3,s1+s2+s3))
```

The function main suspends its execution

s1 = a = 1.2

s2 = b = - 3.2

s3 = c = -4.5

6.4 Functions and Parameters: The Exciting Details

Control of flow:

```
def main():

    a = float(input("Please input a decimal number:"))
    b = float(input("Please input another decimal number:"))
    c = float(input("Please input the last decimal number:"))

    product(a,b,c)

    sum_(a,b,c)

...
```

```
def sum_(s1,s2,s3):
    print("The sum of three numbers {0} +
{1} + {2} = {3}".format(s1,s2,s3,s1+s2+s3))
```

```
The sum of three numbers 1.2 + -3.2 + -4.5 = -6.5
```

6.4 Functions and Parameters: The Exciting Details

Control of flow:

```
def main():

    a = float(input("Please input a decimal number:"))
    b = float(input("Please input another decimal number:"))
    c = float(input("Please input the last decimal number:"))

    product(a,b,c)

→   sum_(a,b,c)

...
```

```
Please input a decimal number:1.2
Please input another decimal number:-3.2
Please input the last decimal number:-4.5
The product of three numbers 1.2 x -3.2 x -4.5 = 17.28
The sum of three numbers 1.2 + -3.2 + -4.5 = -6.5
```

6.5 Functions That Return Values

Functions may return a value to the caller

use `return` statement for that

6.5 Functions That Return Values

Functions may return a value to the caller

use `return` statement for that

Example: Let's write a function that squares a given value and returns it to the caller.

```
def main():
    x = float(input("enter a decimal number:'))
    b = square(x)
    print(b)

def square(a)
    return a*a

main()
```

see program square.py

6.5 Functions That Return Values

Example: Let's write a program that takes a sentence from a file (file name is provided by the user), removes all spaces and stores the result into a new file called *result.txt.* Operation of space removal should be defined as a separate function that returns a string.

6.5 Functions That Return Values

Example: Let's write a program that takes a sentence from a file (file name is provided by the user), removes all spaces and stores the result into a new file called *result.txt.* Operation of space removal should be defined as a separate function that returns a string.

```
def removeSpaces(sentence):
split sentence into a list by space as separator
join the list with no separator
return the new string
```

6.5 Functions That Return Values

Example: Let's write a program that takes a sentence from a file (file name is provided by the user), removes all spaces and stores the result into a new file called *result.txt.* Operation of space removal should be defined as a separate function that returns a string.

```
def main(sentence):
get the file name (fname)
open the file
read one line
call function removeSpaces - that will remove the
spaces and will return the result as a string
output the result to the screen and to the file
'result.txt'
```

6.5 Functions That Return Values

Example: Let's write a program that takes a sentence from a file (file name is provided by the user), removes all spaces and stores the result into a new file called *result.txt.* Operation of space removal should be defined as a separate function that returns a string.

see program removeSpaces.py

6.5 Functions That Return Values

Functions in Python can return more than one value!

6.5 Functions That Return Values

Functions in Python can return more than one value!

Example: let's define a function that given two integer values returns their sum, product and difference.

In Python interpreter:

```
>>> def it(a,b):
      return a+b, a*b, a-b

>>> it(1,2)
(3, 2, -1)
```

6.5 Functions That Return Values

Example: Let’s write a program that draws a triangle based on three user's mouse clicks, then finds the perimeter of the rectangle and displays it in the same graphics window, where rectangle is drawn. The distance calculation between two points will be defined as a function.

6.5 Functions That Return Values

Example: Let's write a program that draws a triangle based on three user's mouse clicks, then finds the perimeter of the rectangle and displays it in the same graphics window, where rectangle is drawn. The distance calculation between two points will be defined as a function.

```
def distance(point1, point2):
    dx = point1.getX() - point2.getX()
    dy = point1.getY() - point2.getY()
    return math.sqrt(dx*dx+dy*dy)
```

6.5 Functions That Return Values

Example: Let's write a program that draws a triangle based on three user's mouse clicks, then finds the perimeter of the rectangle and displays it in the same graphics window, where rectangle is drawn. The distance calculation between two points will be defined as a function.

```
def main():
    create graphics window
    notify the user
    get three mouse clicks
    create and display the triangle (Polygon)
    calculate the perimeter = side1+side2+side3
    display the perimeter value
```

see program **triangle.py**