LECTURER: TAI LE QUY

Introduction to Programming with Python

INTRODUCTION TO PROGRAMMING WITH PYTHON
TOPIC OUTLINE

Modules and Packages

6

UNIT 4

Functions

STUDY GOALS

- Define functions and understand their usefulness
- Understand Python scope rules for variables and functions
- Know how to define and use a function with arguments

EXPLAIN SIMPLY

1. How does one define/declare a function and why is that useful?
2. What is Python’s scope?
3. How does one make a function to return values?

Definition

A function is a self-contained block of codes that performs a specific task. A function must be defined as such which can then be called by other parts of the program. To define a function, one uses the `def` keyword. The block of codes within the `def` will be executed when the function is called.

Syntax: `def <function_name>:`

Example:

```
def greetings():
    print('Hello Python!')

greetings()
```

Keyword `def` signifies that this code block is a function. The name of the function is `greetings`

The function is called by its name `greetings()`. The function code block will then be executed.

Why functions?

Functions are useful for several reasons.

Reusability

- Once a function is defined, it can be used anywhere in the program and even by other programs

Maintainability

- When changes to specific tasks are required, one only needs to modify the function without changing the code elsewhere

Readability

- Since functions encapsulate specific tasks, they make the code more readable.

Definition	When a variable or function is not defined in a program, the variable or function is said to be out of scope. The program will return an error.

Examples

Out of scope

```
greetings()
def greetings():
    print('Hello Python!')
```

At this point of execution, the function `greeting()` has not been defined. It is therefore out of scope. The program returns an error.

In scope

```
def greetings():
    print('Hello Python!')
greetings()
```

At this point of execution, the function `greeting()` has been defined. It is therefore in scope. The program returns no error.

Source of the text: Lutz, 2013.

Variables in functions

Variables defined within a function are only in scope within that function. Outside that function, those variables are out of scope. This will result an error.

```
def greetings():
    my_variable = 15
    print('my_variable = ',my_variable)
```

`my_variable` has been defined. It is in scope inside the function.

```
greetings()
print('my_variable = ',my_variable)
```

Outside the function, `my_variable` is out of scope thus resulting an error.

Note that a variable that is declared before a function is, however, in scope inside the function!

```
my_variable = 15
def greetings():
    print('my_variable = ',my_variable)
```

`my_variable` has been defined and is in scope inside the function.

Source of the text: Lutz, 2013.

`global` keyword

One uses the `global` keyword inside the function to make a variable accessible outside a function when declaring the variable.

```
def greetings():
    global my_variable
    my_variable = 15
    print('my_variable inside = ',my_variable)

greetings()
print('my_variable outside = ',my_variable)
```

By declaring `my_variable` as `global`, it is now in scope not only inside the function but also outside

After the function is called, `my_variable` is in scope outside the function

Definition

Arguments are data that are passed into a function when it is called. To take on arguments, a function must be declared with the parameters (corresponding to the arguments) inside the parentheses after the function name.

Example:

```
def greetings(first_name):
    print('Hello, ',first_name,'!')

greetings('Jane')
```

Function is defined with the parameter `first_name`.

Argument `'Jane'` is passed into the `greetings` function.

Note: 1. A function can have many arguments which are separated by commas.
2. Arguments are in scope inside the function.

Passing arguments

Positional arguments

The position of arguments in the function call responds directly to the position defined in the function.

Example:

```
def personal_info(name, age, position):
   print('name:',name,' age:',age,' position:',position)

personal_info('Jane', 25, 'Senior manager')
```

Source of the text: Lutz, 2013.

Passing arguments

Named arguments

The arguments are labelled by the name in the function definition. Since the arguments are named, they do not have to be in the same position as in the function definition.

Example:

```
def personal_info(name, age, position):
   print('name:',name,' age:',age,' position:',position)

personal_info(age=25,position='Senior manager',name='Jane')
```

Source of the text: Lutz, 2013.

Passing arguments

Default arguments

The arguments can take on default values. This is done in the definition of the function.

Example:

```
def personal_info(name='John', age=20, position='Developer'):
   print('name:',name,' age:',age,' position:',position)

personal_info(age=25,position='Senior manager',name='Jane')
personal_info('Frank',32,'CFO')
personal_info(position='Director',name='Maria')
personal_info()
```

`return` keyword	While a function can take on arguments, it can also return values. This can be done by using the `return` keyword inside the function.

Example:

```
def rectangle_area(length, width):
   return length * width

length = float(input('Length: '))
width = float(input('Width: '))
print('The area is: ', rectangle_area(length, width))
```

Note: 1. Any data type e.g., string, lists, tuple, dictionaries etc can be returned.
2. Any codes after the return statement inside a function will not be executed.

REVIEW STUDY GOALS

- Define functions and understand their usefulness
- Understand Python scope rules for variables and functions
- Know how to define and use a function with arguments

SESSION 4

TRANSFER TASK

1. Inspect the following code. What are the outputs?

```
def personal_info(name='John', age=20, position='Developer'):
    print('name:',name,' age:',age,' position:',position)

personal_info(age=25,position='Senior manager',name='Jane')
personal_info('Frank',32,'CFO')
personal_info(position='Director',name='Maria')
personal_info()
```

2. Inspect the following code. What are the outputs?

```
secret = 15
def funct():
      secret = 28
      print('Level 1 (before)=',secret)
      def nested_funct():
            global secret
            secret = 47
            print('Level 2=', secret)
      nested_funct()
      print('Level 1 (after)=', secret)

funct()
print('Level 0=',secret)
```

TRANSFER TASK
PRESENTATION OF THE RESULTS

Please present your results.
The results will be discussed in plenary.

LEARNING CONTROL QUESTIONS

1. Why are functions useful?

LEARNING CONTROL QUESTIONS

2. What is scope?

3. Why is there an error when trying to run the following code?

```
a=int(input())
b=int(input())
if c < a + b:
    print('Less than a + b')
    a = b = c = 1

print(a,b,c)
```

LIST OF SOURCES

Lutz, M. (2013). Introducing Python Object Types. *Learning Python* (5th ed.). O'Reilly.