Lecture 8 – More practice defining functions, functions can call other functions, the scope of variables

At the end of this lecture, students should be able to:
• write functions which perform a task
• understand that a function can call another function
• understand the scope of variable
• always use excellent function names and variable names to ensure that the purpose of the function is clear

Recap

From lecture 7
• functions which accept arguments and return values can be defined
• calls to functions which have been defined cause the code inside the function to be executed
• we must use excellent function names and variable names to ensure that the purpose of the function is clear
• each function performs one task

```python
def get_dice_total():
    dice1 = random.randrange(1, 7)
    dice2 = random.randrange(1, 7)
    return dice1 + dice2

def get_discount_price(price):
    discount_price = price * 0.95
    return discount_price

dice_throw = get_dice_total()
full_price = 345.67
final_price = get_discount_price(full_price)
```

Syntax of a Python function

A Python function has the following syntax:

```
def function_name(comma_separated_parameters):
    statements in the function
    return value_to_be_returned
```

- 'def' Function name Function parameters colon
- Indentation (either 1 tab or 4 spaces)
- 'return' Return value Statements in the body of the function.
Functions with no return statement

If a function does not need to return a result, then an optional `return` statement can be used as the last statement of the function (see lines 4 and 8).

```python
def display_welcome(name):
    message = "Welcome **" + name + "**"
    print(message)
    return

def display_cost(dollars, cents):
    cost_str = "Cost is $" + str(dollars) + ":" + str(cents)
    print(cost_str)
    return

display_welcome("Sam")
print()
display_cost(15, 35)
```

Welcome **Sam**

Cost is $15:35

See slide 14 of lecture 4:

None is a special value which can be assigned to a variable and it means that the variable is not referencing (pointing to) any object.

Functions with no return statement

In Python, functions which do not explicitly return any value, in fact return the value `None` by default.

```python
def display_welcome(name):
    message = "Welcome **" + name + "**"
    print(message)

def display_cost(dollars, cents):
    cost_str = "Cost is $" + str(dollars) + ":" + str(cents)
    print(cost_str)

display_welcome("Sam")
display_cost(15, 35)
```

Welcome **Sam**

None

Cost is $15:35

See slide 14 of lecture 4:

None is a special value which can be assigned to a variable and it means that the variable is not referencing (pointing to) any object.

Functions with no parameters

Functions may not need to have any parameters inside the round brackets. If the function does not need to receive any information in order to do its job then there will not be any parameters in its parameter list.
Python programs are structured through indentation

- All programming languages use blocks of code and in all programming languages, it is desirable that blocks of code be indented (this is a style requirement, not a language requirement). This principle makes it easier to read and understand code.
- In Python, indentation of blocks of code is a language requirement, not a matter of style. All statements belonging to the same block of code have the same indentation, i.e., the statements within a block line up vertically. The block ends at a less indented line or at the end of the program. If a block has to be more deeply nested, it is simply indented further to the right.

```python
import blah
n = blahblahblah
n = n + blahblahblah
blahblahblahblah:
    c1 = blahblahblah
    c2 = blahblahblah
    blahblahblah:
        blahblahblah
        blahblahblah
print("The end")
```

Python code is structured through indentation. Below is a diagram showing the indentation of a Python program which contains no function definitions.

```
def display_intro()
    message = "Game of Nim"
    print(message)

def display_winner_details(winner, score):
    message = "*** " + winner.upper() + " (" + str(score) + ") ***"
    print(message)

display_intro()
display_winner_details("Joe Li", 56)
```

Game of Nim

```plaintext
*** JOE LI (56) ***
```
Python - colon

Each statement marking the beginning of an indented block ends with a colon. Below is a diagram showing the indentation of a Python program which contains no function definitions.

```
Block 1
Block 2
Block 2 continued
Block 3
Block 1 continued
```

Python - indentation

Each statement marking the beginning of an indented block ends with a colon. Below is a diagram showing the indentation of a Python program which contains two function definitions.

```
def ...:
    Function body

def ...:
    Function body
```

Python – program execution

A Python program starts executing at the first unindented statement (line 7 in the code below).

When the Python interpreter comes across statements (other than def or import ... or a few other keywords) which are written in the left-most column of the program, it will start the program by executing these statements.

```
def display_intro():
    message = "Game of Nim"
    print(message)

def display_winner_details(winner, score):
    message = "*** " + winner.upper() + " (" + str(score) + ") ***"
    print(message)

display_intro()
print()
display_winner_details("Jo Li", 56)
```
Local variables and their scope

When you set the value of a variable inside a function, the Python interpreter creates a local variable with that name. In the following example, the variables: message, author, length and symbols are local variables defined inside the display_intro() function.

In a function, local variables exist from the moment they are set (used) until the end of the function block inside which they are used. For example the variable, author, exists from line 3 to line 9.

```
def display_intro():
    message = "Game of Nim"
    author = "by Adriana Ferraro"
    length = max(len(message), len(author))
    symbols = "*" * length
    print(symbols)
    print(message)
    print(author)
    print(symbols)

display_intro()
```

Variables – out of scope

When you try to use a variable which is out of scope, the interpreter will display an error message:

```
def display_intro():
    message = "Game of Nim"
    author = "by Adriana Ferraro"
    length = max(len(message), len(author))
    symbols = "*" * length
    print(symbols)
    print(message)
    print(author)
    print(symbols)

display_intro()
print(author)
```

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

Exercise

Complete the output of the following program.

```
def display_intro():
    message = "Game of Nim by Adriana Ferraro"
    length = len(message)
    symbols = "*" * length
    print(symbols)
    print(message)

display_intro()
message = "bye bye!"
```

```
Game of Nim by Adriana Ferraro
Game of Nim by Adriana Ferraro
```

The scope of parameters

Parameters are the variables which are listed in the function header.

- The scope of parameters is the same as for local variables, i.e., they exist from the moment they are set (at the beginning of the function) to the end of the function block inside which they are listed, i.e., until the end of the function definition. In the example below the parameters, winner and score, exist from line 1 to line 4.

```
def display_winner_details(winner, score):
    message = "*** " + winner.upper() + " (" + str(score) + ") ***"
    print(message)

display_winner_details("Joe Li", 56)
```

```
*** JOE LI (56) ***
```
Example with four function calls

```python
def get_winner_message(name):
    message = '*** Game of Nim: " + name + " is the winner ***
    return message

def display_winner_details(score, winner_message):
    message = '(" + str(score) + " points"
    number_of_blanks = (len(winner_message) - len(message)) // 2
    blanks = " " * number_of_blanks
    print(winner_message)
    print(blanks + message)

message = get_winner_message("Sam")
display_winner_details(66, message)
print()
display_winner_details("Helen", 178)
```

*** Game of Nim: Sam is the winner ***
(66 points)

*** Game of Nim: Helen is the winner ***
(178 points)

**Exercise**

Complete the get_discount() function which returns the discount amount (a float rounded to 2 decimal places). The function is passed two parameters, the amount and the discount rate (an integer %).

```python
def get_discount(amount, rate):
    discount_message = "Discount: $" + str(get_discount(234, 5))
    print(discount_message)
    discount_message = "Discount: $" + str(get_discount(125, 15))
    print(discount_message)
```

Discount: $11.7
Discount: $18.75

**Exercise**

Complete the get_discount_message() function which returns a string made up of the rate of discount, the string "% Discount: $", and the discount amount. The function has two parameters, the discount amount and the rate of discount (a number).

```python
def get_discount_message(discount, rate):
    discount_message = get_discount_message(11.7, 5)
    print(discount_message)
    discount_message = get_discount_message(98.55, 15)
    print(discount_message)
```

5% Discount: $11.7
15% Discount: $98.55
Exercise

Complete the print_docket() function which prints the sales docket information (the format should be as shown in the example output shown). The function is passed two arguments, the price and the discount rate (an int %). Your function code MUST make a call to both functions: get_discount() and get_discount_message().

```python
def get_discount(amount, rate):
    # code from slide 23

def get_discount_message(discount, rate):
    # code from slide 24

def print_docket(price, rate):

    print_docket(234, 5)
    print_docket(657, 15)
```

Exercise

The following program prompts the user for a number of items to be packaged. Each box can hold 10 items. Any left over items require an extra box. The first 6 boxes cost $8 each and any boxes above the first 6, cost $5 each. The program executes as shown in the example outputs below. Design the functions needed to write this program and write the main code for this program.

```python
# write the main code below
items_per_box = 10
```

Exercise

From the previous slide.

```python
Enter number of items: 20
Items: 20
Boxes needed: 2
Cost: $16

Enter number of items: 65
Items: 65
Boxes needed: 7
Cost: $53

Enter number of items: 102
Items: 102
Boxes needed: 11
Cost: $73
```

Show the errors

The following program has two errors. What are the errors? Write a correction for each error.

The desired output is shown below the program.

```python
def display_winner_details(winner, score):
    message = "*** " + winner.upper() + " (" + score + ") ***"
    print(message)

score = score + 50
display_winner_details("Joe Li", score)
```

```python
*** JOE LI (50) ***
50
```
In a Python program:

- functions can be used to perform various tasks
- a function can make calls to other functions
- the scope of variable needs to be understood
- It is important to always use excellent function names and variable names to ensure that the purpose of the function is clear

Examples of Python features used in this lecture

```python
def display_welcome(name):
    message = "Welcome **" + name + " **"
    print(message)
    return

def display_intro(name):
    local_variable = "Game of Nim"
    local_variable = local_variable + "by " + name
    print(local_variable)

def display_menu():
    print("1. Option 1")
    print("2. Option 2")
    print("3. exit")

display_menu()
display_welcome("Sam")
display_intro("Adriana Ferraro")
```