CHAPTER 3: FUNCTIONS

Johan Van Bauwel
E-mail: Johan.vanbauwel@thomasmore.be
Office: A114

OVERVIEW

- Defining a function
- Calling a function
- Function arguments
- Return statements
- None value
- Global statements
- Exception handling
- Keyword arguments and print()
- Local and global scope

DEFINING A FUNCTION

- *def* keyword
- Followed by a colon :
- Mind the indentation

```
def hello():
        print('Howdy!')
        print('Howdy!!')
        print('Hello there.')
```

CALLING A FUNCTION

- Function name, followed by parentheses

```
hello()
hello()
hello()
```

FUNCTION ARGUMENTS

- Local variable between parentheses

```
def hello(name):
        print('Hello' + name)
```

- Function calls, executing function hello(name)

```
hello('Alice')
hello('Bob')
```

FUNCTION ARGUMENTS

- Result

Hello Alice
Hello Bob

- Name is a local variable! It can’t be called outside the function hello()!

RETURN STATEMENTS

- Value that a function call evaluates to

```
import random
def getAnswer(answernumber)
    if answernumber == 1:
        return 'number is 1'
    elif answernumber == 2:
        return 'number is 2'
```

RETURN STATEMENTS

```
r = random.randint(1,20)
fortune = getAnswer(r)
print(fortune)
```

- Alternative

```
print(getAnswer(random.randint(1,20)))
```

NONE VALUE

- In case of absent value
- ex. print() does not return a value, but displays text on screen
  - return None

```
spam = print('Hello')
#spam = return value from print() = None
print(spam == None)

Hello!
True
```

NONE VALUE

- len() returns a value

```
spam = len('Hello')
#spam = return value from print() = None
print(spam)
print(spam == None)

5
False
```

KEYWORD ARGUMENTS AND PRINT()

- Python automatically adds newline character at the end of each line
- To disable this, use the *end* keyword

```
print('Hello', end='') #keyword end changes newline to ''
print('World')

HelloWorld
```

KEYWORD ARGUMENTS AND PRINT()

- Use *sep* to choose a separation symbol

```
print('Hello', 'World', sep=',')
```

```
Hello,World
```

LOCAL AND GLOBAL SCOPE

```
def spam():
        eggs=31337
spam()
print(eggs)
```

- will not work → eggs is local variable
- local scopes cannot use variables in other local scopes

LOCAL AND GLOBAL SCOPE

- Global variables can be read in local scopes

```
def spam():
        print(eggs)
eggs = 42
spam()
```

- result = 42

LOCAL AND GLOBAL SCOPE

- you can use the same name for a local and global variable
  - changing local value won't affect global value

```
name = ‘Castor‘
def changeName():
    name = ‘Pollux’
    print(name)
changeName()                    Pollux
print(name)                     Castor
changeName()                    Pollux
```

GLOBAL STATEMENTS

- Keyword *global* is used to change global variable from within function

```
name = 'Castor'
def changeName():
    global name
    name = 'Pollux'
    print(name)
changeName()                    Pollux
print(name)                     Pollux
changeName()                    Pollux
```

EXCEPTION HANDLING

- Error?

```
def divideBy(number):
        return 42/number

divideBy(0)
```

```
ZeroDivisionError: division by zero
```

EXCEPTION HANDLING

- Display message when error detected

```
def divideBy(number)
        try:                                    #code with potential error
                return 42/number
        except ZeroDivisionError:       #errorhandling
                print('Error: Invalid argument.')
```

SHORT PROGRAM: GUESS THE NUMBER

```
# This is a guess the number game.
import random
secretNumber = random.randint(1, 20)
print('I am thinking of a number between 1 and 20.')

# Ask the player to guess 6 times.
for guessesTaken in range(1, 7):
    print('Take a guess.')
    guess = int(input())
```

SHORT PROGRAM: GUESS THE NUMBER

```
if guess < secretNumber:
        print('Your guess is too low.')
elif guess > secretNumber:
        print('Your guess is too high.')
else:
        break    # This condition is the correct guess!

if guess == secretNumber:
    print('Good job! You guessed my number in ' + str(guessesTaken) + ' guesses!')
else:
    print('Nope. The number I was thinking of was ' + str(secretNumber))
```

QUIZ

1. Why are functions advantageous to have in your programs?
2. When does the code in a function execute: when the function is defined or when the function is called?
3. What statement creates a function?
4. What is the difference between a function and a function call?
5. How many global scopes are there in a Python program? How many local scopes?
6. What happens to variables in a local scope when the function call returns?

QUIZ

7. What is a return value? Can a return value be part of an expression?
8. If a function does not has a return statement, what is the return value of a call to that function?
9. How can you force a variable in a function to refer to the global variable?
10. What is the data type of None?

QUIZ

11. If you had a function named bacon() in a module named spam, how would you call it after importing spam?
12. How can you prevent a program from crashing when it gets an error?
13. What goes in the try clause? What goes in the except clause?

Q & A

- Questions?