

Introduction to Programming



UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Functions

Introduction to Functions

CSC1



Hussein Suleman
Department of Computer Science

Introduction to Functions

Hussein Suleman
Department of Computer Science
University of Cape Town

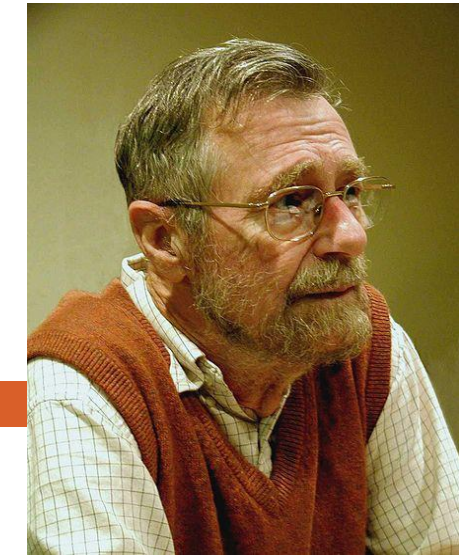


UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Problem 1 Introduction

- Write a program to print out the reverse of a sentence.
 - For example:
 - Computer becomes retupmoC
- Use first principles - i.e., process the string character-by-character.
- Use functions to make your program readable/modular.

“Go to considered harmful”



“I was terribly frightened. ...

I would sit all evening silently staring at the white walls in our living room. Finally, one night at half past two, my wife collected me weeping on the carpet in that room.

From that moment I realized that something had to be done. I started writing 'Notes on Structured Programming' for therapeutic reasons. When that text was written, I knew what I had to do, and I knew how I was going to attack it

‘Notes on Structured Programming’ broke on the computing community with the force of a revolution. For several years, people had known that something was seriously wrong with software.”

Edsger Dijkstra (1930-2002)

Dutch Computer Scientist, speaking on Structured Programming in 1968

Olson, Steve. "Sage of software." Science '84, vol. 5, Jan.-Feb. 1984, pp. 74+. Gale Academic.

Function

- A function is a named block of statements that can be executed/called within a program.
- We have already used some functions:
 - `print, eval, round, ...`
- Python stops what it is doing, runs the function, then continues from where it stopped.
- Functions enable reuse and modularity of code.
- Functions help us to write longer/more complex programs.

So far:

```
# reverse an integer without using strings
number = eval (input ("Enter a number: "))
reverse = 0
while number > 0:
    digit = number % 10
    number = int(number / 10)
    reverse = reverse * 10 + digit
print (reverse)
```

Function - example

```
# our program

# reverse an integer without using strings
number = eval(input("Enter a number: "))
reverse = 0
while number > 0:
    digit = number % 10
    number = int(number / 10)
    reverse = reverse * 10 + digit
print(reverse)
```

```
# somewhere else

def input (someString):
    #function to read from standard input

def eval (aStringNumber):
    #converts string to number

def int(aString or aFloat):
    #returns an integer using floor function

def print(oneOrMoreStrings):
    #outputs the strings to standard output
```


Function Definition / Use

- Functions can be defined and used in any order, as long as they are used after definition.

- To define a function:

```
def some_function () :  
    statement1  
    statement2
```

...

- To use/call/invoke a function:

```
some_function ()
```

Code refactoring

- Functions can refactor code to avoid duplication

```
print ("Welcome")
print ("to")
print ("CS1")
print ("Welcome")
print ("to")
print ("CS2")
print ("Welcome")
print ("to")
print ("CS3")
```

```
def welcome():
    print ("Welcome")
    print ("to")
welcome ()
print ("CS1")
welcome ()
print ("CS2")
welcome ()
print ("CS3")
```

Poll

Why do we use functions when we write programs?

- A – to reuse code
- B – to write large, modular programs
- C – to increase code readability
- D – all of the above

Solution

Why do we use functions when we write programs?

- A – to reuse code
- B – to write large, modular programs
- C – to increase code readability
- D – all of the above



Poll

What Python keyword is used to define a function?

- A – `func`
- B – `def`
- C – `function`
- D – `define`

Solution

What Python keyword is used to define a function?

- A – `func`
- B – `def` ✓
- C – `function`
- D – `define`

Exercise

- Write a function to print the sequence of integers 1-10, each on a new line.
- Execute the function once.



UNIVERSITY OF CAPE TOWN

IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

IT | SCHOOL OF IT

Unless otherwise stated, all materials are copyright of the University of Cape Town

© University of Cape Town

Introduction to Programming



UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Functions

Parameters

CSC1



Hussein Suleman
Department of Computer Science

Parameters

Hussein Suleman
Department of Computer Science
University of Cape Town



UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Parameters

- Parameters allow variation in function behaviour

```
print ("Welcome")  
print ("to")  
print ("CS1")  
print ("Welcome")  
print ("to")  
print ("CS2")  
print ("Welcome")  
print ("to")  
print ("CS3")
```

```
def welcome(grp):  
    print ("Welcome")  
    print ("to")  
    print (grp)  
  
welcome ("CS1")  
welcome ("CS2")  
welcome ("CS3")
```

Parameters

- Every function can have a list of parameters in its definition.
 - called the **formal parameters**
- Whenever the function is called/invoked a value must be provided for each of the formal parameters
 - called the **actual parameters** or **arguments**
- Within the function body, the parameters can be used like variables.

Formal and Actual Parameters

```
def some_function (a, b, c):  
    print (a)  
    print (b+c)
```

formal parameters
(or just “parameters”)

```
some_function (12, 23, 34)
```

actual parameters
or “arguments”

Pass-By-Value

- Only a copy of the value of a parameter is ever sent to a function.
- So if there is an original variable, it cannot be changed by the function changing the parameter.

Pass-By-Value

```
def some_function (a):  
    a=a+1  
    print (a)
```

```
b = 12  
some_function (b)  
print (b)
```

output

```
13  
12
```

Return Values

- Functions can return values just like mathematical functions.
- Use the **return** statement with an expression.
- Can be used anywhere in function and will return immediately.

Return Values

```
def square (x):  
    return x*x
```

```
y = square (12)  
print (y)
```

output

144

Poll

What is a formal parameter?

- A – name used within function
- B – value passed to function
- C – name associated with arguments
- D – formal names used anywhere in program

Solution

What is a formal parameter?

- A – name used within function
- B – value passed to function
- C – name associated with arguments ✓
- D – formal names used anywhere in program

Poll

When does the name **a** refer to an actual parameter?

- A – `return a`
- B – `a=dosomething()`
- C – `def dosomething(a)`
- D – `dosomething(a)`

Solution

When does the name **a** refer to an actual parameter?

- A – `return a`
- B – `a=dosomething()`
- C – `def dosomething(a)`
- D – `dosomething(a)`



Exercise

- Write a function to print the sequence of integers 1-n, each on a new line, depending on the value of the parameter n.
- Execute the function for values 3, 5 and 7.



UNIVERSITY OF CAPE TOWN

IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

IT | SCHOOL OF IT

Unless otherwise stated, all materials are copyright of the University of Cape Town

© University of Cape Town

Introduction to Programming



UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Functions

Identifier Scope

CSC1



Hussein Suleman
Department of Computer Science

Identifier Scope

Hussein Suleman
Department of Computer Science
University of Cape Town



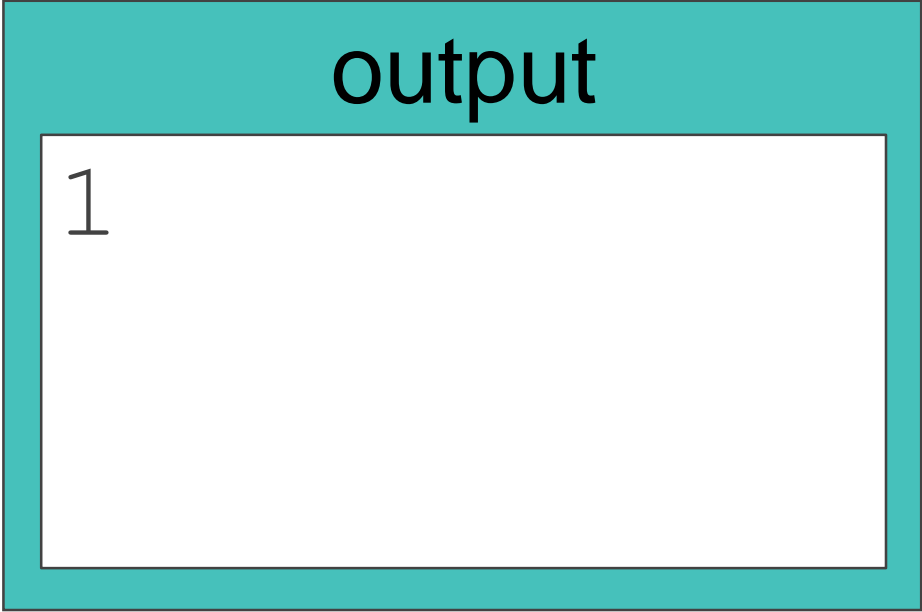
UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Scope and Local Variables

- **Scope** refers to where a variable can be used (accessed or changed).
- New variables can be created and used within functions but they disappear when the function ends.
 - called **local variables**
- Limiting scope:
 - Prevents errors in one part of a program affecting other parts.
 - Helps programmers to manage large programs.

Scope and Local Variables

```
def some_function () :  
    a = 1  
    print (a)  
  
some_function ()
```



output

1

Scope and Local Variables

- Local variable names (and parameters) that are the same as global variable names temporarily hide the global variables.

Scope and Local Variables

```
def some_function (a,c):  
    a = 3  
    b = 3  
    print (a,b)  
  
a = 1  
b = 2  
  
some_function (1,2)  
print (a,b)
```

output

```
3 3  
1 2
```

Global Variables

- Global variables are not within any function.
- Global variables can be accessed but not changed.
- Use the **global** statement to allow changes to a global variable.

Global Variables

```
def some_function (a):  
    global b  
    b = 4  
    a = 3  
  
b = 2  
some_function (b)  
print (b)
```

output

4


Poll

Why is scope important?

- A – avoid having to track lots of variables
- B – prevent programming errors
- C – so we know which variable a name refers to
- D – All of the above

Solution

Why is scope important?

- A – avoid having to track lots of variables
- B – prevent programming errors
- C – so we know which variable a name refers to
- D – All of the above 



UNIVERSITY OF CAPE TOWN

IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

IT | SCHOOL OF IT

Unless otherwise stated, all materials are copyright of the University of Cape Town

© University of Cape Town

Introduction to Programming



UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Functions

Problem 1

CSC1



Hussein Suleman
Department of Computer Science

Problem 1

Hussein Suleman
Department of Computer Science
University of Cape Town



UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Problem 1

- Write a program to print out the reverse of a sentence.
- For example:

```
Enter a sentence: Computer  
retupmoC
```
- Use first principles - i.e., process the string character-by-character.
- Use functions to make your program readable/modular.



UNIVERSITY OF CAPE TOWN

IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

IT | SCHOOL OF IT

Unless otherwise stated, all materials are copyright of the University of Cape Town

© University of Cape Town

Introduction to Programming



UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Functions

Modular Code

CSC1



Hussein Suleman
Department of Computer Science

Modular Code

Hussein Suleman
Department of Computer Science
University of Cape Town



UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

docstring

- Functions should be documented by specifying their purpose in a string immediately after the header.
- It is recommended that you use `"""` (triple quotes - for multi-line strings) for all **docstrings**.
- Use `func.__doc__` to check the docstring.

docstring

```
def cube (x):  
    """Return the cube of x."""  
    return x*x*x  
  
def square (x):  
    """Return the square of x.  
    x can be any numerical value"""  
    return x*x
```

nested functions

- Functions can be composed similarly to mathematical functions.

```
def cube (x):  
    return x*x*x  
  
def square (x):  
    return x*x  
  
def power (a, b):  
    return a**b  
  
print (power (cube (2), square (2)))
```

main function

- Common practice is to wrap a program into a function called "main", then invoke this function to execute the program.

```
# cube program
def cube (x):
    return x*x*x
def main ():
    print (cube (2))
main()
```

Writing your own modules

- Any file with functions can be imported.
- Check `__name__` variable
 - if it is `"__main__"`, then this file was executed
 - otherwise, this file was imported

Writing your own modules

```
# cube module
def cube (x):
    return x*x*x
def main ():
    print (cube (2))
if __name__=="__main__":
    main()
```

```
# test cube module

import a

print (a.cube(3))
```

Exercise

- Modify the program to reverse a string such that it can be used as a separate reusable module.
- Use a main function that asks for input and prints the reversed string, as before.



UNIVERSITY OF CAPE TOWN

IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

IT | SCHOOL OF IT

Unless otherwise stated, all materials are copyright of the University of Cape Town

© University of Cape Town

Introduction to Programming



UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Functions

Problem 2-3

CSC1



Hussein Suleman
Department of Computer Science

Problem 2-3

Hussein Suleman
Department of Computer Science
University of Cape Town



UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Problem 2

- Write a function that prints a text heading inside a box made up of + characters.
 - Store this function as a module.
 - Use an optional parameter to specify the padding within the box.

output

```
+++++++  
+ Hello +  
+++++++
```

Default values for parameters

- Functions can have zero or more parameters with default values in their definition.
- All parameters with default values must be at the end of the parameter list.
 - e.g., once you have a default value, all subsequent parameters must have a default value as well.

Default values for parameters

- Evaluated at the time of function definition, not invocation.
- Whenever the function is called/invoked, the arguments with default values are optional.
- Within the function body, parameters are still used as variables.

Recall: Formal Parameters

```
def some_function (a, b, c):  
    print (a)  
    print (b+c)
```

formal parameters

```
some_function (12, 23, 34)
```

actual parameters

Default values for formal parameters

```
def some_function (a, b, c=10):  
    print (a)  
    print (b+c)
```

formal parameter
with default value

```
some_function (12, 23, 34)
```

optional parameter

```
some_function (12, 23)
```

Problem 2

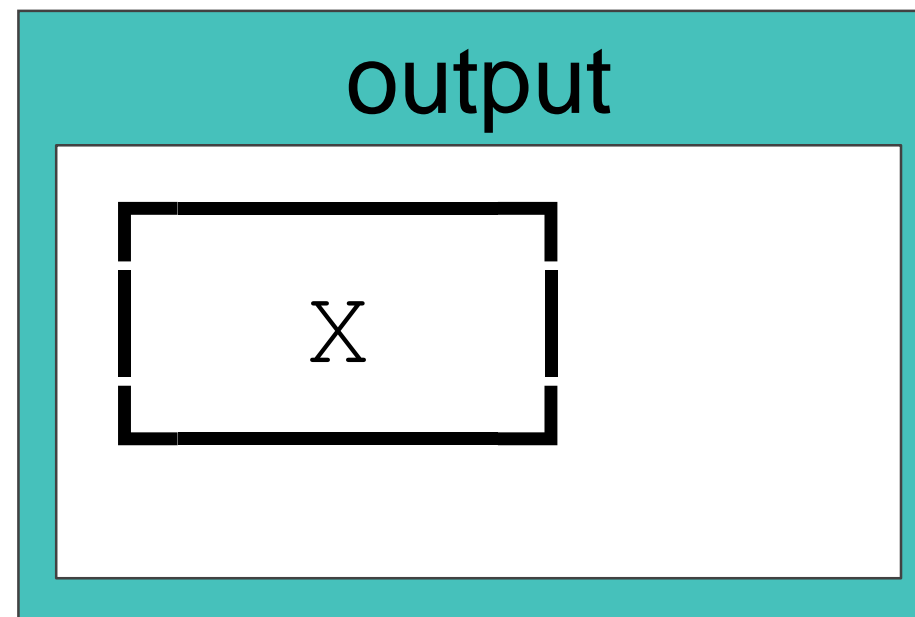
- Write a function that prints a text heading inside a box made up of + characters.
 - Store this function as a module.
 - Use an optional parameter to specify the padding within the box.

output

```
+++++++  
+ Hello +  
+++++++
```

Problem 3

- There are special Unicode characters for box corners and edges.
 - For example “\u250F” is the top left corner symbol.
- Modify your heading program to use these characters instead of “+”.



```
# Unicode box test
print ("\u250F\u2501\u2501\u2501\u2513\n")
print ("\u2503 X \u2503\n\u2517\u2501\u2501\u2501\u251B\n")
```



UNIVERSITY OF CAPE TOWN

IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

IT | SCHOOL OF IT

Unless otherwise stated, all materials are copyright of the University of Cape Town

© University of Cape Town