



**UCT Department of Computer Science**  
**Computer Science 1015F**

# Functions



*Aslam Safla*

*<aslam@cs.uct.ac.za>*

*(thanks to Hussein Suleman <hussein@cs.uct.ac.za>)*

# 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.



# 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.



# 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" )
```

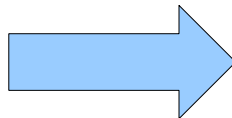
```
print ( "to" )
```

```
print ( "CS2" )
```

```
print ( "Welcome" )
```

```
print ( "to" )
```

```
print ( "CS3" )
```



```
def welcome():
```

```
    print ( "Welcome" )
```

```
    print ( "to" )
```

```
welcome ( )
```

```
print ( "CS1" )
```

```
welcome ( )
```

```
print ( "CS2" )
```

```
welcome ( )
```

```
print ( "CS3" )
```



# 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

---

formal parameters

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

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

actual parameters



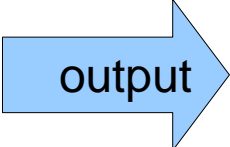
# Pass-By-Value

---

- ❑ 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.

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

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

 13  
12



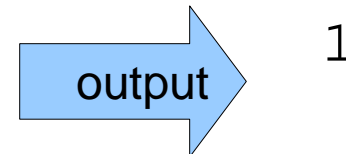
# Scope and Local Variables 1/2

---

- ▣ New variables can be created and used within functions but they disappear when the function ends.
- called **local variables**

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

```
some_function ()
```



# Scope and Local Variables 2/2

---

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



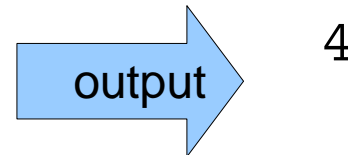
# Global Variables

---

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

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

```
b = 2  
some_function (b)  
print (b)
```



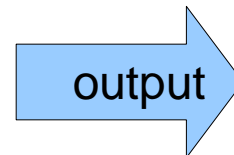
# Return Values

---

- ▣ 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.

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

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



144



# Problem 1

---

- 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.



# 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.

```
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

```
# 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))
```



# Problem 2

---

- ❑ Convert the freewifi program to use functions, with all the best practices for using functions.



# Problem 3 Intro 1/2

---

□ Write an application to tell what country or body of water the International Space Station is over right now. Use best practices for functions.



□ Extend your application to give the next time the ISS will fly over South Africa



# Problem 3 Intro 2/2

---

- <http://api.open-notify.org/iss-now.json> Will give the current latitude and longitude of the International Space Station
- <http://www.geonames.org/export/web-services.html#countrycode> Can help us find a country name from a latitude and longitude
- <http://www.geonames.org/export/web-services.html#ocean> Can help us find the ocean name from a latitude and longitude



# International Space Station

---



- ❑ First component launched in 1998
- ❑ Modular space station
- ❑ Largest man-made body in orbit
- ❑ Can often be seen with the naked eye from Earth
- ❑ Science!



# Default values for parameters

---

- ❑ Functions can have zero or more parameter 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.
- ❑ Evaluated at the time of function definition, not invocation
- ❑ Whenever the function is called/invoked, the arguments with default values are **optional**.
- ❑ Within the fn body, parameters still used as variables.



# Recall: Formal Parameters

---

formal parameters

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

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

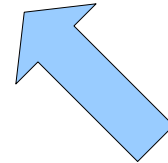
actual parameters



# Default values for formal parameters

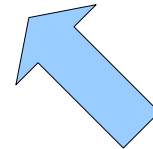
formal parameter  
with default value

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



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

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



optional parameter



# Problem 3

---

□ Write an application to tell what country or body of water the International Space Station is over right now. Use best practices for functions.

□

□ Extend your application to give the next time the ISS will fly over South Africa

