ISYS90088 Introduction to Application Development

Week09 & 10 – Introduction to Functions

Dr Thomas Christy

Sem 2, 2017

Functions

- What's a function? It is a group of statements that exist within a program for the purpose of performing a task
  - (much like in Maths) functions take a set of input values, perform some calculation based on them, and optionally return a value
  - you have already seen and used many functions by this stage, e.g.: str(), len(), sqr(), min(), max()...
-

Functions: usefulness

- Simpler code
- Code re-use
- Better testing
- Faster development
- Easier facilitation of teamwork

Functions: defining and calling

- The code for a function is known as a function definition. To execute a function, you write statement that calls it.
- In order to define a function, we need:
  - A function name (following same conventions as other variable names)
  - (optionally) a list of input variables
  - (optionally) a UNIQUE output object (via return)

Functions: defining and calling

- Function names should be:

✓ Descriptive enough so that anyone reading your code can reasonably guess what the function does (prefer to use verbs)

✓ Rules for naming:

- ➢ Cannot use python key words
- ➢ Cannot contain spaces
- ➢ First character must be letter; after the first character, you may use letters, digits, underscore
- ➢ Case sensitive

Functions: defining

Basic syntax :

```
def <function_NAME>(INPUTLIST):
            statement
            statement
            statement
```

Example:

```
def message():
   print(‘this is a simple case’)
```

Functions: calling

- To execute a function, you must call the function.

```
<function_name> ()
```

Example:

```
def message():
    print('this is a simple case')


message()
```

Functions: main() and functions

```
# a function to perform something
#the main function
def main():
    message()

def message():
     print('this is a simple case')



#execute main function
main()
```

Functions: simple examples

#This is a simple example to illustrate a function call

```
def main():
    print("I have a message for you")
    message()
    print('Good bye')

def message():
    print('this is Antonette')
    print('can you hear me?')

#call main program
main()
```

Example: two functions

Write two functions: one called melbourne() and another called canberra(). The functions must accept from the user the number of SME's in each of these two cities. It must then print the SME's in each of the cities.

```
def main():
    melbourne()
    canberra()
#defining the two function
def XXX:
    small_medium =  XXXX
    XXX
def XXX:
    small_medium =  XXXX
    XXX
XXX
```

Example: two functions

Write two functions: one called melbourne() and another called canberra(). The functions must accept from the user the number of SME's in each of these two cities. It must then print the SME's in each of the cities.

```
def main():
    melbourne()
    canberra()
#defining the two function
def melbourne():
    small_medium =  10000
    print('the SME's in Melbourne', small_medium)
def canberra():
    small_medium =  6500
    print('the SME's in Canberra', small_medium)
main()
```

Scope and local variables

Local variable: is created inside a function and cannot be accesses by statements that are outside the function.

- Different functions within a program can have same variable names since the other functions cannot see or use each others *local variables*.
- A variable's *scope* is the part of a program in which the variable may be accessed.
- A local variable cannot be accessed by code that appears inside a function at a point before the variable has been created.

Examples : 1 – what happens here?

```
def main():
    melbourne()
    canberra()

#defining the functions
def melbourne():
    birds = 1000
    print('melbourne has', birds, 'birds')

def canberra():
    birds = 870
    print('canberra has', birds, 'birds')

# calling the main function
```

Examples: 2 – what's the problem here?

```
def main():
    get_name()
    print('hello', name) #causes an error

def get_name():
    name = input('enter your name:')

main()

#another scope - issue. A local variable cannot be
accessed by code below that appears???? Why????.

def bad_function():
    print("the vakue is", val) # causes an error
    val = 99
```

Functions: passing arguments to functions

- A argument is any piece of data that is *passed into* a function when the function is called.
- A parameter is a variable that *receives* an argument that is passed into a function.

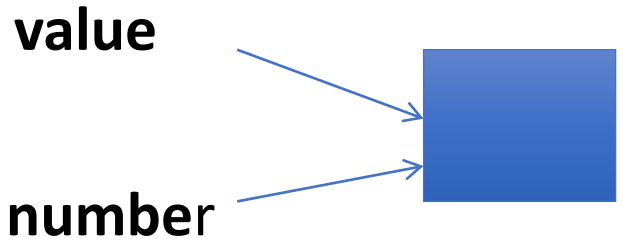
- Many times we send across pieces of information (data) into a function and tasks are performed within the function.
- And many times information is passed back from a function to the main that called this function using a **`return`** statement .

Example: how to pass values

```
def main():
    value = int(input('enter a number:'))
    show_double(value)

def show_double(number):
    result = number * 2
    print(result)

main()
```



Functions: simple examples

write a function to print the number of digits in a number

```
def print_digits(n):
        s = str(abs(n))
        print (len(s) - ('.'  in  s))
```

write a function to convert from Celsius to Fahrenheit:

```
def print_C2F(n):
        print(9*n/5 + 32)
```

Functions: simple examples

Print the number of digits in a number

```
def print_digits(n):
        s = str(abs(n))
        print (len(s) - ('.'  in  s))

def main():
    v = float(input('enter a value:'))
    print_digits(v)

main()
```

Convert from Celsius to Fahrenheit:

```
def print_C2F(n):
      print(9*n/5 + 32)
```

Now write the main function that calls this function?

Passing multiple arguments

Example: Write a program that demonstrates a function that accepts two arguments and then displays their sum.

```
def main():
    print ('the sum of 12 and 45 is')
    show_sum(12, 45)


def show_sum (num1, num2):
    result = num1 + num2
    print(result)
main()
```

Parameter list - The values 12 and 45 are arguments that are passed by *position* to the corresponding parameter variables in the function

Passing multiple arguments

Example: Write a program that demonstrates a function that accepts two arguments and then displays their sum.

```
def main():
    print ('the sum of 12 and 45 is')
    show_sum(12, 45)


def show_sum (num1, num2):
    result = num1 + num2
    print(result)
main()
```

Parameter list - The values 12 and 45 are arguments that are passed by *position* to the corresponding parameter variables in the function