Python Subroutines

Anastasis Oulas
Evangelos Pafilis
Jacques Lagnel

Why use subroutines?

- As programs and algorithms get more complicated the tend to incorporate pieces of code which have been already written and utilized elsewhere.
- One way to overcome the hassle of copy-pasting and maintain a better control over the structure of our program is to write **subroutines**

Characteristics of subroutines

- Has an input and an output
- Is independent from one another
- Should be short and concise

Advantages of subroutines

- Makes the development of the algorithm and program easier and less prone to error
- Makes the understanding and correction of an algorithm or program easier
- Allows for a significant saving of time as an already written subroutine can be called from elsewhere in the algorithm.
- Play a crucial role in the expansion of programming languages by allowing for new subroutines to be implemented and stored in libraries.

Parameters

- In order to execute a subroutine it has to be ***called*** or invoked from the **main program** or from another subroutine.
- When calling a subroutine certain variables can be passed into the subroutine as **parameters.**

Some in-built subroutines you have used already!

- print(x)
- print(x, y, z)
- str1.**find**(site) # returns an integer (position)
- re.**findall**(regEx,seq) # returns a list (matched regEx patterns)

Creating a subroutine

- A subroutine is created with the command "**def**" followed by the name we as the programmers give to our subroutine
- Subroutines are usually defined at the **start** of a program
- Example

```
def mysubroutine(parameters):
    .....
    return something
```

Example subroutine

```
def AddTwoNumbersAndPrint(x,y):
    z = x+y
    print(z)
```

Calling a subroutine

- In order to call a subroutine we simply do so by calling it with its name

  …..mysubroutine(…)

- Inside the brackets we add the parameters that will be passed into the subroutine from the main program
  - These may be:
    - Nothing at all (simple subroutine)
    - static value(s)
    - Variable(s) (i.e. Strings, Integers, Dictionaries)
    - Combination of both variables and static values

Example calling a subroutine

```
#-------subroutine-----------------------------------
def AddTwoNumbersAndPrint(x,y):                    #Defining a
    z = x+y                                        subroutine
    print(z)
#-----------end of subroutine-----------------------------



#------main program-----------------------------------
AddTwoNumbersAndPrint(2,5) #call to the subroutine
#-----------end of main program-----------------------
```

Example calling a subroutine

```
#-------subroutine-----------------------------------
def AddTwoNumbersAndPrint(x,y):
    z = x+y
    print(z)
#-----------end of subroutine-----------------------

#------main program----------------------------------
x = 2
y = 5
AddTwoNumbersAndPrint(x,y) #call to the subroutine
#-----------end of main program----------------------
```

#Defining a subroutine

but variables x and y also exist in the subroutine as ***local variables!!!***

variables x and y exist in the main program

Notes about the code

- In python subroutines are always defined at the top of the program before the main program starts
- Calling a subroutine is done so using its name from the main program or from another subroutine
  - i.e. mySubroutine(…)
- Parameters can be passed into the subroutine from the main program using brackets
  - i.e. mySubroutine(x,y)
- Parameters in the main program and the subroutine may or may not have the same name(s)
- However changing their values in the subroutine does not change their values in the main program – they are ***local*** to the subroutine

Return a value from the subroutine to the main program

```
#-------subroutine-----------------------------------
def AddTwoNumbersAndPrint(x,y):
    z = x+y
    Return(z)
#----------end of subroutine-----------------------------
```

#Defining a subroutine

```
#------main program-----------------------------------
z = AddTwoNumbersAndPrint(2,5) #      to the subroutine
print(z)
#----------end of main program----------------------
```

By adding a return statement in the subroutine you can return any local variable from the subroutine to the main program.

Example of subroutine that returns the minimum of a list of integers

```
#-------subroutine-------------------------------------
def calculate_min(list):
    min = list[0]
    for i in range(1,len(list)):
        if(list[i] < min:
            min = list[i]
    return min
#----------end of subroutine-------------------------------

#------main program----------------------------------------
alist  = [12,-5,54,73,1, 178, 91]
min = calculate_min(alist)
#calling subroutine parsing alist to its parameters
print(min)
#----------end of main program---------------------------
```

Example of subroutine that returns a dictionary

```
#-------subroutine----------------------------------------
def Count_DNA(Seq):
        Counts = {} # initialize a dictionary
        countA = 0
        countC = 0
        countG = 0
        countT = 0
        for i in range(len(Seq)):
                if(Seq[i] == 'A'):
                        countA = countA+1
                        Counts['A'] = countA
                elif(Seq[i] == 'C'):
                        countC = countC+1
                        Counts['C'] = countC
                elif(Seq[i] == 'G'):
                        countG = countG+1
                        Counts['G'] = countG
                else:
                        countT = countT+1
                        Counts['T'] = countT
        return Counts
#----------end of subroutine-------------------------------

#------main program----------------------------------------
Seq = 'ACGCGACGACGACGACTACCCCCGACTA'
#calling subroutine parsing String (Seq) as parameter into subroutine
Counts = Count_DNA(Seq)
print(Counts)
#----------end of main program--------------------------
```

Defining the subroutine

Calling the subroutine from the main program