Subroutines

Subroutines are named blocks of code which exists as sub-programs within a program

They can be “**called**” from other parts of the program, the code executes and values can be “**returned**” to the main program

They are used for specialised tasks or for tasks which need to be repeated a number of times

Subroutines

Subroutines are examples of the decomposition of an algorithm

Decomposition occurs when a problem is broken down into a series of sub-problems

Each sub-problem may well be a subroutine

Subroutines

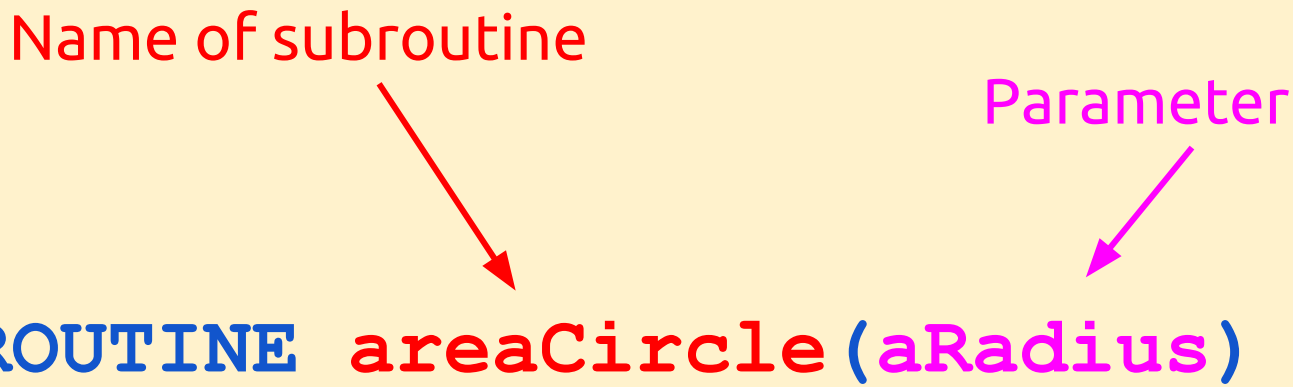
In Python, subroutines are implemented using **functions**

Subroutines are usually placed at the top of all program code

```
def areaRectangle():
```

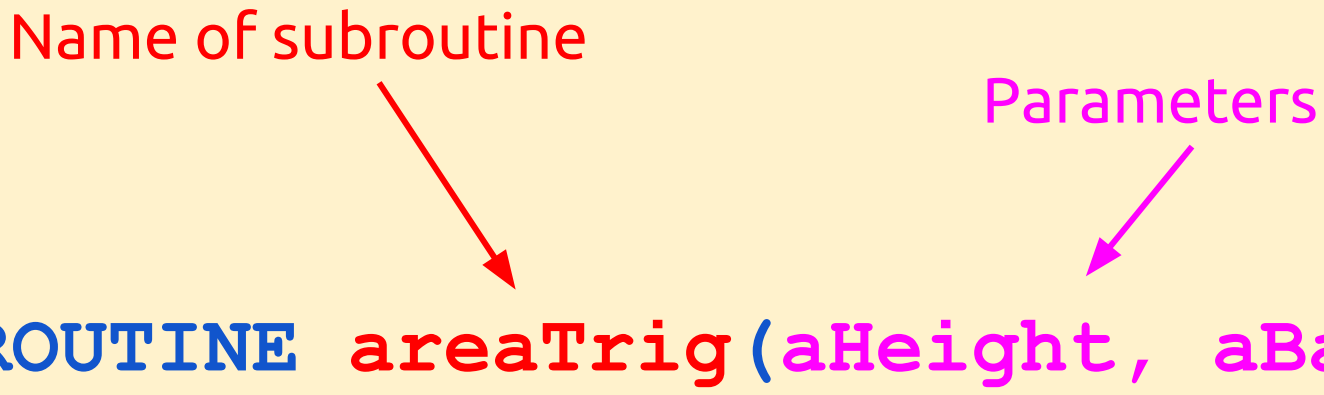
Subroutines

```
SUBROUTINE areaCircle(aRadius)
  area <- aRadius * aRadius * pi
  return area
ENDSUBROUTINE
```

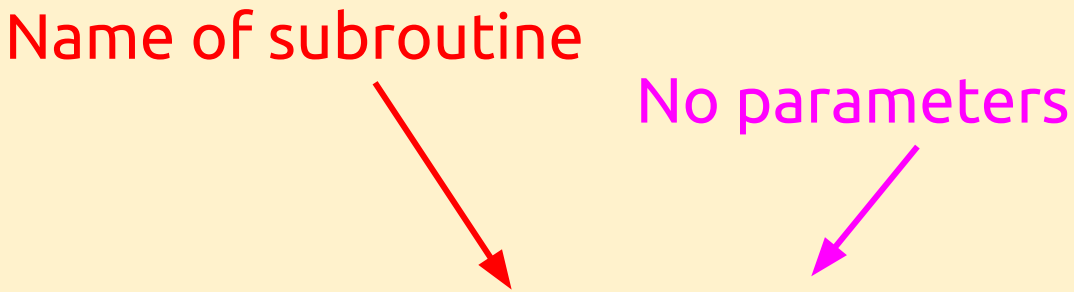


A **parameter** is a variable passed from the main program to the subroutine where it can be used

These can then be used within the subroutine



Subroutines can have more than one parameter




```
SUBROUTINE areaTrig()
```

Or have no parameters

(in this case the values would probably be entered by the user inside the subroutine)

Brackets are still needed!

SUBROUTINE areaTrig(aHeight, aBase)

To “call” this subroutine from another part of the program you’d write any of:

```
areaTrig (7, 2)   # using values

areaTrig(ht, bs)   # using variables


answer <- areaTrig(ht, 4)

# assigning to variable
```

Subroutines

Subroutines can be called from within other subroutines

ADVANCED: They can even be called from within themselves (a process called recursion)