Learning Objectives

- Describe what a subprogram is
- Explain the benefits of using subprograms
- Know the difference between a procedure and function
- Explain the Create subprograms that use parameters
- Use the subprograms symbol in flowcharts
- Explain the difference between global and local variables

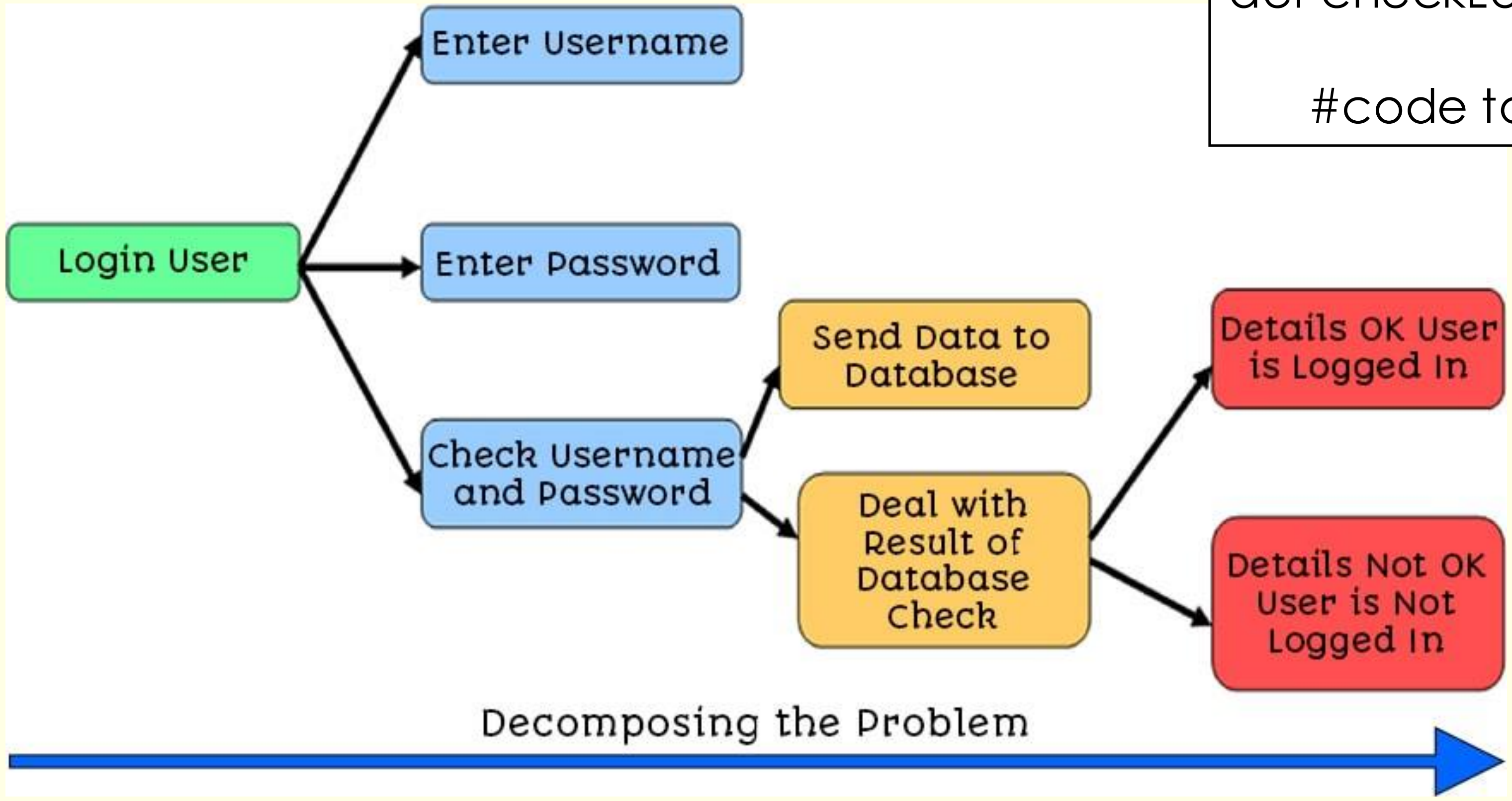
Key Terms

Subroutine/Subprogram	A sequence of instructions to perform a specific task with an identifiable name.
Function	A subroutine that returns a value.
Procedure	A subroutine that executes a block of code when called. It does not return a value.
Parameter	Used in a subroutine to allow values to be passed into them.
Argument	The values held in the brackets of a subroutine call. These are passed into a subroutine via the parameters.
Decomposition	Breaking down a problem into smaller subproblems to make the more manageable.

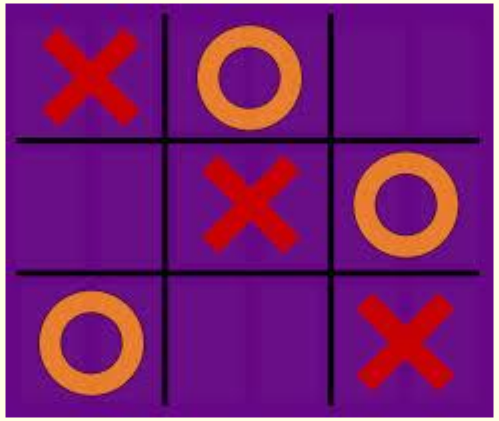
Decomposition of a solution

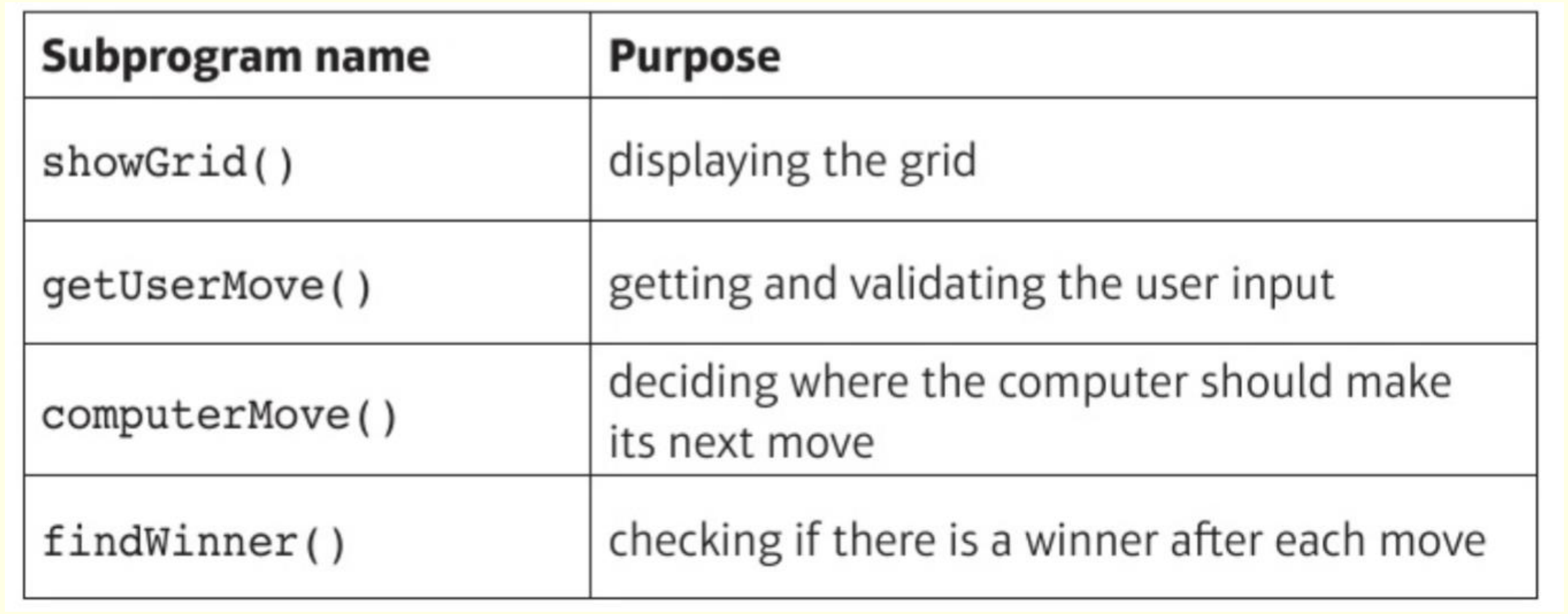
- More complex solutions can be broken down into parts, known as decomposition.
- These parts are implemented by coding them into subprograms.
- Subprograms are blocks of code that perform a specific task.
- You can use built-in subprograms and write your own.

```
def checkLogin(username, password):

    #code to check username and password
```


Login User
Enter Username
Enter Password
Check Username and Password
Send Data to Database
Deal with Result of Database Check
Details OK User is Logged In
Details Not OK User is Not Logged In
Decomposing the Problem

Subprograms for a Noughts and Crosses Game





Subprogram name	Purpose
`showGrid()`	displaying the grid
`getUserMove()`	getting and validating the user input
`computerMove()`	deciding where the computer should make its next move
`findWinner()`	checking if there is a winner after each move

Built in libraries

- You are using pre-written code when you use:
  - **print(), input(), int(), str()**
- **Modules (known as libraries)** are collections of subprograms that are pre-written.
- This makes developing programs much quicker!!
- With most modules we have use the import command at the top of the code to be able to use them, for example:

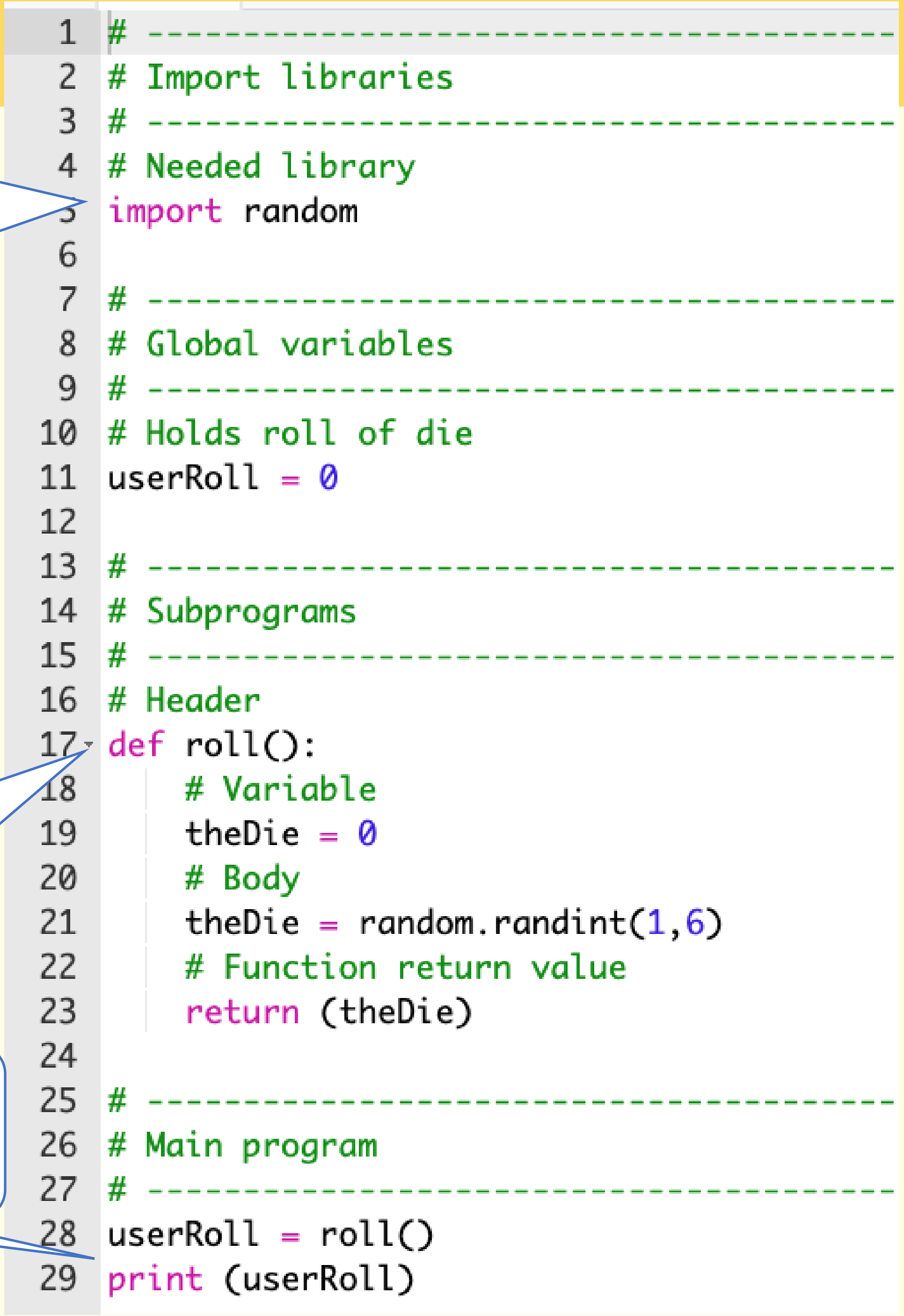
```
import time
time.sleep(2)

import random
random.randint(0,6)
```

Worked example

library subprogram, random.randint()

programmer-written subprogram, roll()

built-in subprogram, print()

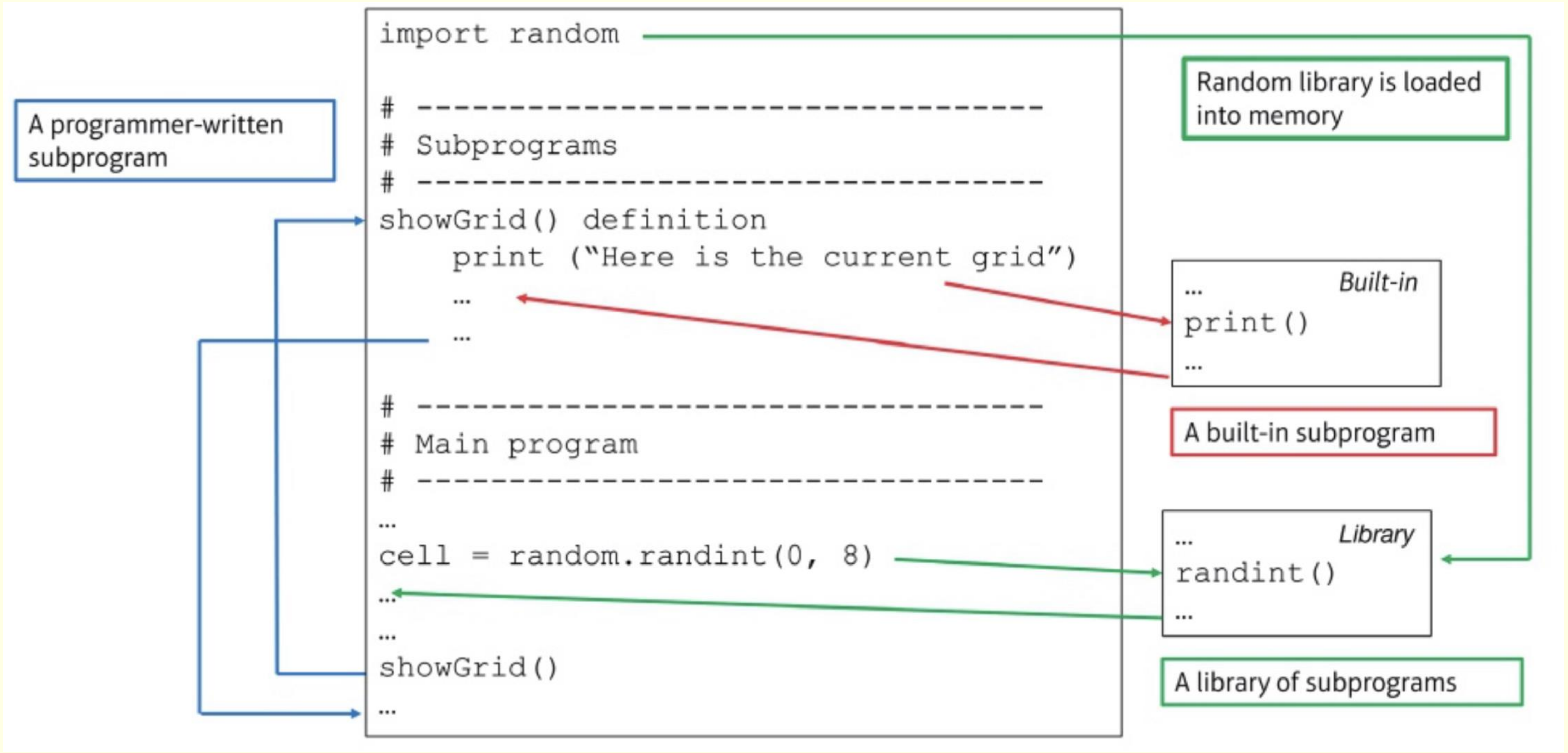


```
# ---------------------------------------
# Import libraries
# ---------------------------------------
# Needed library
import random

# ---------------------------------------
# Global variables
# ---------------------------------------
# Holds roll of die
userRoll = 0

# ---------------------------------------
# Subprograms
# ---------------------------------------
# Header
def roll():
    # Variable
    theDie = 0
    # Body
    theDie = random.randint(1,6)
    # Function return value
    return (theDie)

# ---------------------------------------
# Main program
# ---------------------------------------
userRoll = roll()
print (userRoll)
```

Using subprograms has many advantages

- Easily **reuse** a block of code within a program
- Logical structure so easier to maintain and debug.
- **Reused** by other programmers
- Speeds up development time

Code layout



```
import random

# ----------------------------------
# Subprograms
# ----------------------------------
showGrid() definition
    print ("Here is the current grid")
    …
    …

# ----------------------------------
# Main program
# ----------------------------------
…
cell = random.randint(0, 8)
…
…
showGrid()
…
```

Two types of subprograms

- A **procedure** is a small section of a program that performs a specific task. Procedures can be used repeatedly throughout a program.

- A **function** is also a small section of a program that performs a specific task that can be used repeatedly throughout a program, but functions perform the task and **return a value to the main program.**

How to write a procedure?

- Writing a **procedure** is extremely simple. Every procedure needs:
- a meaningful name
- Any parameters needed – values passed into the procedure to be used in the program code
- the **program** code to perform the task

```
def procedure_name()
    instructions
```

How to write a function?

- Writing a function is extremely simple.
- Every function needs:
  - a meaningful name
  - Any parameters needed – values passed into the procedure to be used in the program code
  - the program code to perform the task
  - a value to return to the main program

```
def function_name():
      instructions
      return something
```

Worked example - Procedure

Parameters are used to pass values into a subprogram and used inside the block of code.

Parameters

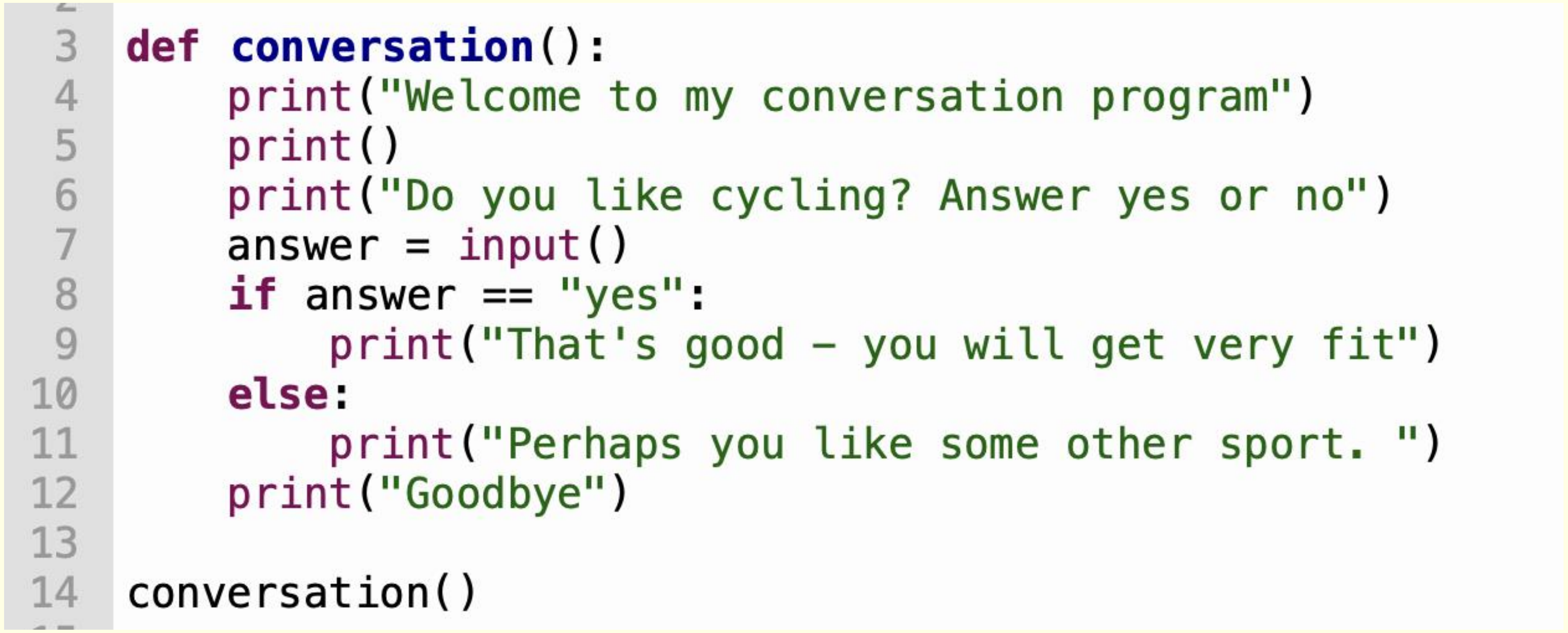
```
def result(name, age, gender):
     print ("My name is" +name)
     print ("My age is" + str(age))
    print("My gender is" + gender)


result("Sam", 13, "male")
```

Call the procedure

Arguments

Another example

This is an example of a subprogram known as a **procedure**.

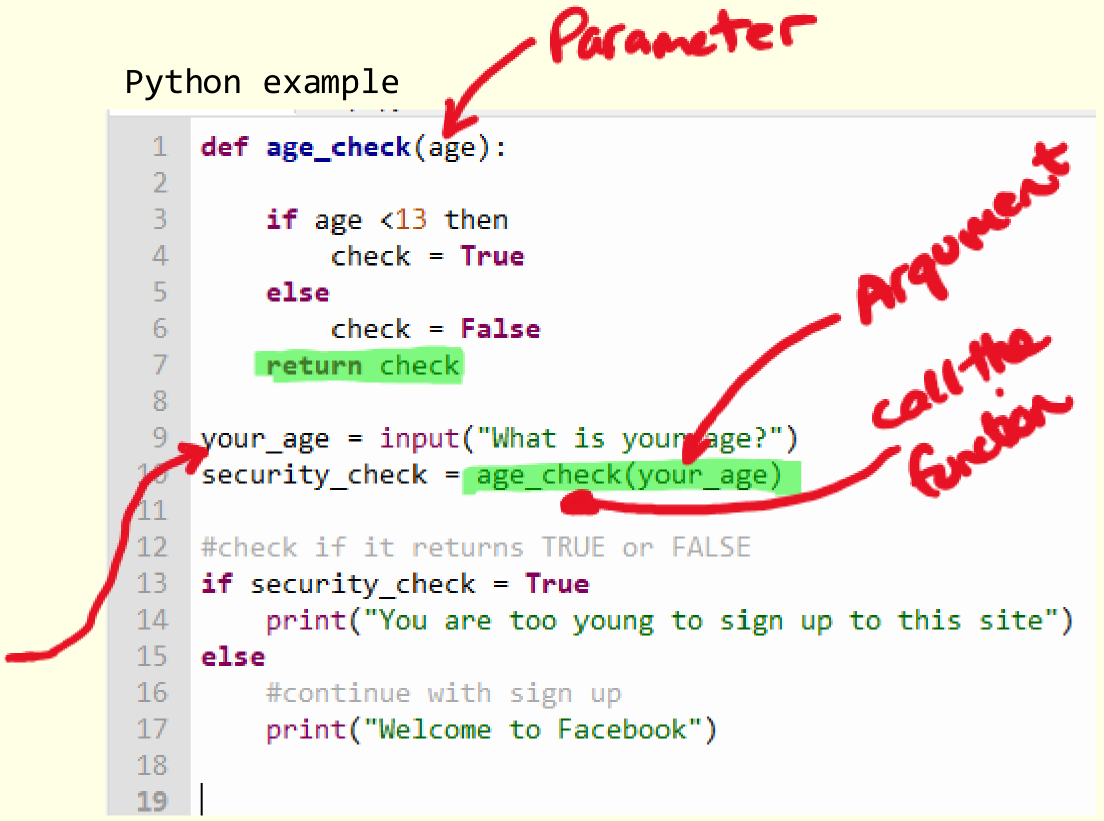


```python
def conversation():
    print("Welcome to my conversation program")
    print()
    print("Do you like cycling? Answer yes or no")
    answer = input()
    if answer == "yes":
        print("That's good - you will get very fit")
    else:
        print("Perhaps you like some other sport. ")
    print("Goodbye")

conversation()
```

Worked example:

- The return statement is used to specify the value to be returned to the main program.
- You must always make a variable to store the returned value

Python example



```
def age_check(age):

    if age <13 then
        check = True
    else
        check = False
    return check

your_age = input("What is your age?")
security_check = age_check(your_age)

#check if it returns TRUE or FALSE
if security_check = True
    print("You are too young to sign up to this site")
else
    #continue with sign up
    print("Welcome to Facebook")

```

Function – returns data

Check if a number is an even number

```
def checkifeven(num)
    check = False
    if   num MOD 2 = 0 then
       check = True
    return check
#use a variable to store the returned value

check = checkifeven(5)
print(check)
```

What will this return

```
check = checkifeven(12)
print(check)
```

What will this return?

Why?

Breaks down / decomposes / modularises the problem / program / structures the program

…making it easier to design/create/test

…each subroutine can be tested separately

Allows for abstraction / removes complexity

…subprograms can be used by programmers who do not need to understand how they work

Reuse code (in different programs)

…quicker to develop (new) programs

…build on existing work / use of a library of subroutines

Easier to maintain

…as code is easier to understand/read

…as code is shorter

Avoid repetition of code (in the same program)

…makes program shorter / smaller

… subprogram called instead of copying/pasting.

… quicker to develop (new) programs

…to work on different subprogram at the same time / develop separately

Easier to debug

…as code is shorter

…same bugs will not have been copied to other areas of the program.

Work can be split up in a team

…to suit developers' skill set

Flowcharting a subprogram

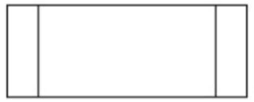


Figure 1.7.3 The symbol for a subprogram in a flowchart

Flowchart

In a flowchart, a subprogram is represented by the symbol shown in Figure 1.7.3.

Fully documenting a complex solution using flowcharts can be cumbersome, and it's quite easy to get lost. Figure 1.7.4 shows what part of the noughts and crosses game might look like using subprograms.

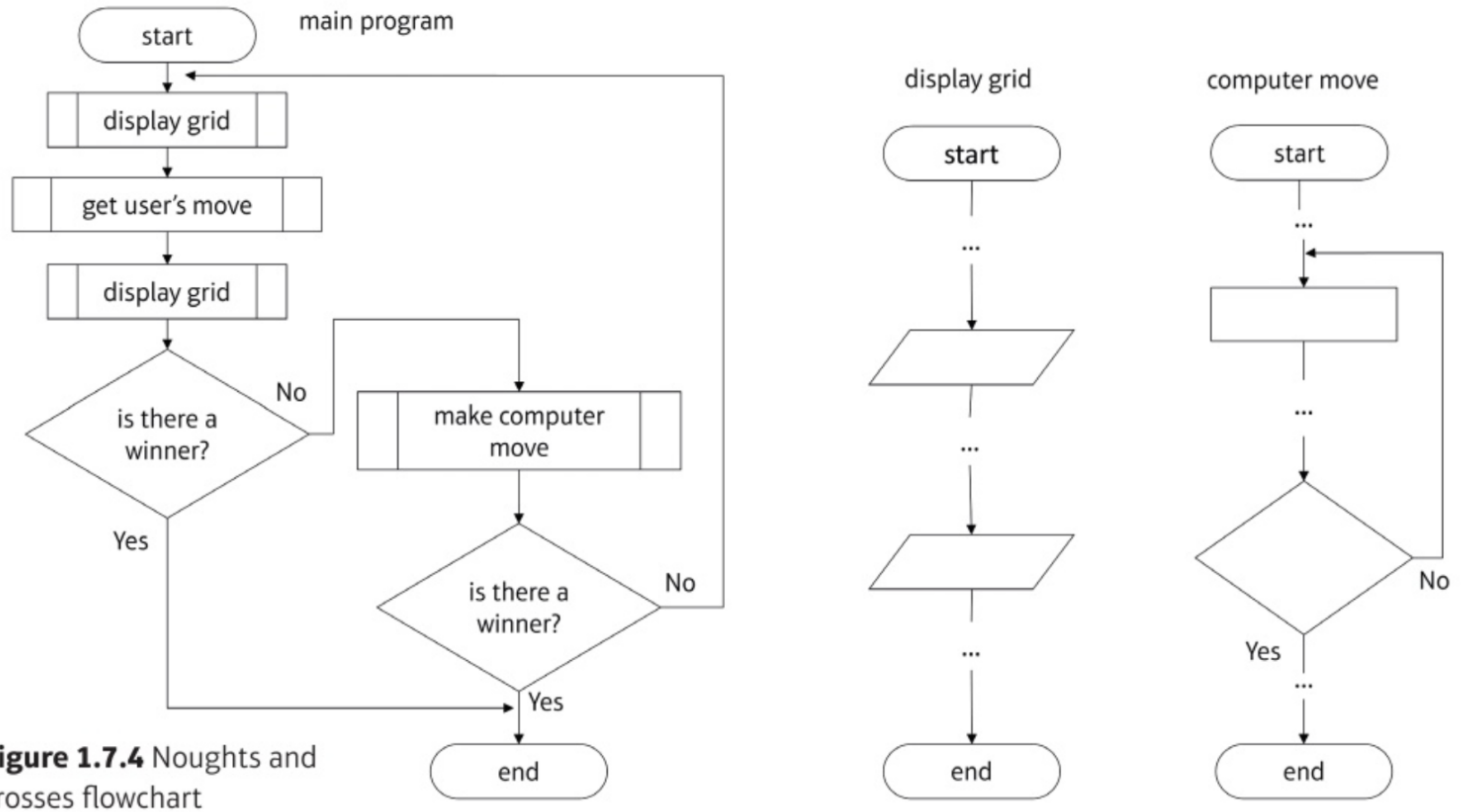
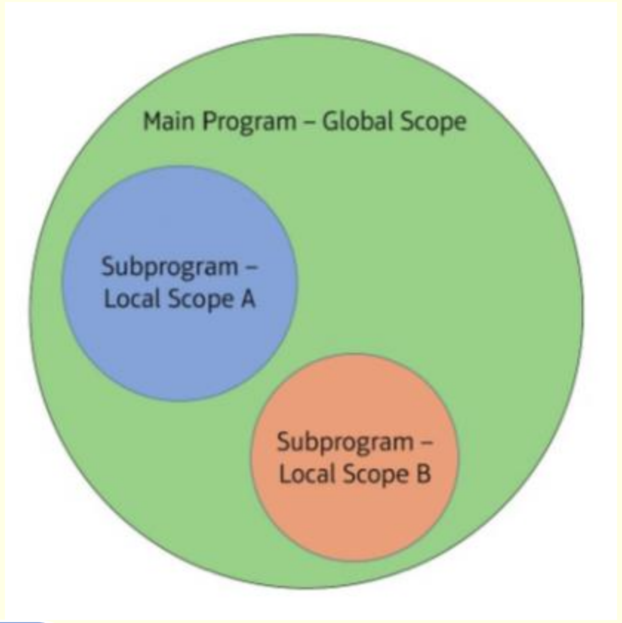


Figure 1.7.4 Noughts and crosses flowchart

Local variables, global variables and scope


Main Program – Global Scope
Subprogram – Local Scope A
Subprogram – Local Scope B

Local and global variables have different scope.

A **global variable** is declared at the beginning of a program and is available throughout the program

A **local variable** is declared within a sub program and it is only available with that section of code and is destroyed when the subprogram exits