Objective 8:

Understand subroutines, procedures and functions

In this objective you learn how to divide your program into more manageable sections to make it easier to read and reduce the need for entering repeating code.

Tasks

1. Try entering the following program and see what happens:

```
#Global variables accessed by all subroutines
MaxNoOfStars=0
NoOfStars=0
NoOfSpaces=0

#Set the size of the pyramid of stars
def InitialiseNoOfSpacesAndStars():
    global NoOfSpaces, MaxNoOfStars, NoOfStars
    MaxNoOfStars=int(input("How many stars at the base (1-40)? "))
    #Calculate spaces needed from tip
    NoOfSpaces = MaxNoOfStars // 2
    #Set tip of pyramid to one star
    NoOfStars = 1

#Outputs spaces before stars
def OutputLeadingSpaces():
    global NoOfSpaces
    for count in range(NoOfSpaces):
            print(" ",end="")

#Outputs stars
def OutputLineOfStars():
    global NoOfStars
    for count in range(NoOfStars):
        print("*",end="")
    #Move to next line
    print()

#Adjusts number of stars & spaces after output
def AdjustNoOfSpacesAndStars():
    global NoOfSpaces, NoOfStars
    NoOfSpaces = NoOfSpaces - 1
    NoOfStars = NoOfStars + 2

#Main program starts here
InitialiseNoOfSpacesAndStars()
while NoOfStars <= MaxNoOfStars:
    OutputLeadingSpaces()
    OutputLineOfStars()
    AdjustNoOfSpacesAndStars()
```

2. Try entering the following program and see what happens:

```
#Program to output a set of random numbers
import random
#Output random numbers
def outputrandoms(n,m):
    for counter in range(1,n+1):
        randomnum = random.randint(1,m)
        print("Number",counter,"is",randomnum)

#Main program starts here
number=int(input("How many numbers do you want to output? "))
maximum=int(input("What is the maximum number? "))
outputrandoms(number, maximum)
```

There are two ways to share data between subroutines. In the first program, the variables were initialised (given a starting value) outside of any subroutine. That means they are global variables to the program, and their values can be accessed, shared and changed by any subroutine in the program, using the command 'global'.

In the second program the variables 'number' and 'max' are passed into the subroutine outputrandoms. The values of the variables become n and m in the subroutine outputrandoms. The variables n and m are different variables to number and max, but the value stored in number and max was passed into n and m. . In this example, the variables n and m are lost at the end of the outputrandoms subroutine. That means they are local variables to the subroutine.

3. Try entering this program and see what happens:

```
#How functions can be used
#Returns a positive number from subtraction
def floor(a, b):
    f = a - b
    if f > 0:
        return f
    else:
        return 0

#Main program starts here
num1=int(input("Enter the first number: "))
num2=int(input("Enter the second number: "))
print("The floor number is:",floor(num1, num2))
```

4. Change the function so it receives the length of two sides of a right angled triangle, and returns the length of the hypotenuse (h2 = o2+a2)

Objective 8: Key learning points

Understand subroutines, procedures and functions

- Subroutines are **sections of code** to break a longer programs into smaller pieces. Thereby making them easier to read and more manageable for teams of programmers to work together on one program.
- Subroutines are also known as **procedures**.
- **Functions** carry out small tasks on data by taking one or more **parameters** and **returning** a result.
- Subroutines/procedures may also take parameters, but do not return a result.
- It is good practice to comment each subroutine or function to explain its purpose.
- Variables declared outside of all subroutines are known as **global variables**. They are available to all subroutines and functions.
- Global variables are not memory efficient since they hold memory for the entire time a program is running. They should therefore be kept to a minimum.
- Variables declared inside a subroutine or function are known as **local variables**. They lose their value when the routine ends, releasing memory.
- Passing variables between subroutines is known as **parameter passing**.

Objective 8: Key words

def…

Example code:
```
def maximum (x,y):
    …
```

Creates a subroutine called max. Two parameters are passed into max, with local variables x and y holding their values.

max would be called with: `x=maximum(6,9)`
6 would be passed as a parameter to variable x. 9 would be passed as a parameter to variable y.

global

Example code:
```
def maximum (x,y):
    global z
    …
```

Includes a global variable in the subroutine.

return

Example code:
```
def maximum (x,y):
    if x>y:
        return x
    else:
        return y
```

Returns a value to the calling routine. E.g. `x = maximum(4,7)` would result in x having the value 7.