Subroutines

Objectives:

To know the advantages of subroutines

To be able to create a subroutine

To be able to call a subroutine

Subroutines/subprograms

A subroutine or subprogram is a block of code that is given a name and can be run from other parts of a program.

It should:

- have a sensible name that describes what it does
- perform a single task

Pseudocode subroutine

```
subroutine  gameInstructions()
    print(“Welcome  to the game”)
    print(“Press z to move left, m to move right”)
    print(“space to fire”)
    print(“Game ends when you lose 3 lives”)
    print(“Press h to see this screen again”)
end subroutine
```

Running the subroutine

You run a subroutine simply by giving its name (we say in programming that we are *calling* the subroutine)

```
gameInstructions()
choice=input(“P to play, H for help Q to quit”)
while choice !=‘Q’
        if choice ==‘p’ then
                playGame()
        end if
        if choice==‘h’ then
                gameInstructions()
        end if
end while
```

Python

```
def gameInstructions():
    print("Welcome  to the game")
    print("Press z to move left, m to move right")
    print("space to fire")
    print("Game ends when you lose 3 lives")
    print("Press h to see this screen again")
```

Create a new program called restaurant.py

In this program, create a subroutine called menu().

It should print a menu with three sections:

- Starters
- Mains
- Desserts

Within each section, print at least 3 dishes!

Run your program, what happens?

Call the menu subroutine 3 times

1. Edit your program so it displays a message welcoming customers to the restaurant and then displays the menu.
2. Display another message asking the customers what they would like to order for their main meal, and display the menu again.
3. Finally display a message asking the customers for their dessert order, and display the menu again.

Your program should only be 6 lines long (not counting the lines you used to create the subroutine itself)

Mini plenary

What are the advantages of subroutines?

- A subroutine can be written once, and used in several places in a program, saving development time.
- Once the subroutine is tested and working, it can be used again and again without fear of introducing any new bugs, leading to more reliable programs.
- Programs that use subroutines are easier to maintain because changes made to the subroutine will not affect the rest of the program.
- Programs are easier to read and follow when made up of carefully named subroutines.

Parameters

When defining or calling a subroutine, you may have noticed the empty brackets.

menu()

These don’t have to be empty. Inside we can put a value called an *argument,* which is passed into the subroutine.

Parameters

In a new program, type the following:

```
def oddOrEven(number):
        if number % 2 == 0:
                print('Even number')
        else:
                print('Odd number')

oddOrEven(2)
oddOrEven(1)
```

Parameters

What is going on here?

```
def oddOrEven(number):
        if number % 2 == 0:
                print('Even number')
        else:
                print('Odd number')

oddOrEven(2)
oddOrEven(1)
```

Modify the restaurant program

Whilst having the menu() subroutine is useful, it would be improved if we could pass in a string to say which part of the menu to print,e.g.

menu(“starters”) # prints just the starters

menu(“mains”) # prints just the mains

menu(“desserts”) # prints just the desserts

Returning values from subroutines

Arguments and parameters are a way of passing data *into* a subroutine. What if you want to return a value from the subroutine to the main program? To do this, we use the keyword return

```
def doubleIt(number):
        return 2 * number
```

Returning values from subprograms

When we call a subprogram that returns a value, we should store the returned value into a variable.

```
def doubleIt(number):
        return 2 * number
doubledNumber=doubleIt(3)
```

Create three subroutines

1. A subroutine that takes in one number and returns the same number multiplied by itself
2. A subroutine that takes in two numbers, and returns the two numbers added together
3. A subroutine that takes in a number and returns True if it is even or False if it is odd