4.1.1 Subroutines

What is a Subroutine?

A **subroutine** is a **named, out-of-line block of code** that performs a specific task and can be called (executed) whenever needed.

Instead of writing the same code repeatedly, a programmer can place it inside a subroutine and call it by using its name.

A subroutine can be:

- A **procedure** – performs a task but does **not return a value**.
- A **function** – performs a task and **returns a value**.

Procedure

```
def greet():
    print("Hello!")
```

Functions

```
def square(number):
    return number * number
```

A subroutine is executed by **calling its name.**

```
greet()
answer = square(5)
```

Parameters and Return Values

Parameters

A subroutine can receive values called **parameters**.

Parameters allow the same subroutine to work with different data.

```
def add(a, b):
    print(a + b)

add(5, 3)
add(10, 20)
```

Return Values

A function can send a value back using return.

```
def calculate_tax(price):
    tax = price * 0.2
    return tax

cost = calculate_tax(100)
print(cost)
```

Why use Subroutines?

Advantages

- A subroutine can be written once and called many times.
- Each subroutine can be tested and debug separately.
- Meaningful subroutine names make programs easier to understand.
- Makes programs easier to maintain and update in the future as fewer changes need to be made.
- Less likely to be errors in the code as the code is just being reused.
- Allows use of recursive techniques.
- Use of local variables reduces side effects due to changes of global variables.