

Introduction to Kotlin Functions

A function is a group of related statements that perform a specific task. Functions are used to break a large program into smaller pieces. For example if we were to write a hypothetical program to draw a person we could break it into small functions such as:

`drawHead()`, `drawTorso()`, `drawArms()` and `drawLegs()`

When we use functions we are able to reuse the functionality defined by a function instead of repeating ourselves throughout the program.

How to write a Kotlin function

- Every Kotlin function is denoted by the `fun` keyword.
- Every Kotlin function must have a name. The name should describe what the function does, e.g `addition()`, `subtraction()`, `print()`, `getSquareRoot()`
- Every Kotlin function must have the invoke operator `()`
- Every Kotlin function must have a body, denoted by opening and closing curly braces `{ }`
- For a Kotlin function to run we must call / invoke it.

To put it all together, some examples:

```
fun add(){
    val a = 32
    var b = 45
    var sum = a + b
}

fun printName(){
    print("Mary Juma")
}

fun modulus(){
    var num1 = 45
    var num2 = 13
    var modulus = num1 % num2
}
```

Invoking/ Calling functions

Having defined the functions above we can call each one of them in the main function or in any other function like this:

```
fun main(){
    add()
}
```

or

```
fun main(){
    printName()
}
```

or even call all of them together

```
fun main(){
    add()
    printName()
    modulus()
}
```

Function Parameters/ Arguments

Take the example of this addition function

```
fun multiply(){
    var a = 12
    var b = 24
    var product = a * b
}
```

It gives us the product of 2 numbers: 12 and 24. Supposing we wanted to multiply a different pair of numbers e.g 34 and 289, we don't want to create another multiplication function. We can make our function reusable by passing the values we want to multiply to it as parameters, a.k.a arguments. e.g

```
fun multiply(num1: Int, num2: Int){
    var product = num1 * num2
}
```

we can now call the addition function with any pair of numbers

```
fun main(){
    multiply(456, 432)
    multiply(112, 89073)
    multiply(1,2)
}
```

Function parameters allow us to reuse functions by invoking them with different values. Another example is

```
fun printName(name: String){
    print("Hello my name is " + name)
}
```

can be invoked with different values like

```
fun main(){
    printName("Anne")
    printName("Judy")
    printName("Musikari")
}
```

 Parameters are variables passed to functions.

Function Return Types and Values

All the functions we have looked at so far perform some tasks and maybe print some output. If we don't add a print statement then the function runs and exits quietly without us noticing what the function has done. Take this example:

```
fun subtract(num1: Int, num2: Int){
    var difference = num1 - num2
}
```

This function will subtract num2 from num1 but we won't be able to observe anything. This is not very useful. Some times we want a function to perform a task and give us the result of the task that has been performed. The above function would now change to:

```
fun subtract(num1: Int, num2: Int): Int{
    var difference = num1 - num2
    return difference
}
```


We are now able to obtain the result of the subtraction like this

```
fun main(){
    var result = subtract(32893, 900)
    print(result) // printing out the result will show 31993
}
```

We are then able to use the result of the subtract function elsewhere, e.g inside another function.
Another example:

```
fun getFullName(firstName: String, lastName: String){
    var fullName = firstName + " " + lastName
    return fullName
}
```

Try this on your own and print the result of the `getFullName()` function.

 The return type of a function is the data type of the result of the function.