Kotlin - Android Dev

Using IntelliJ IDEA for this

- Use fun main() to start of the main block of code in the {}

ex:

fun main( ){

}

- To declare a variable we use the keyword : var

ex: var <var_name>: <datatype> = <value>

- It is not necessary to use the keyword var everytime. Using it only during declaring is enough
- Another way to declare Variables is via → val
- The difference between val and var is the value of var can be reassigned and value of val can't
- It is not necessary to specific datatype after var name
- Using f after a number makes it a float
-

Arithmetic Ops

- / → division, f to be used when needed to get in decimals
- % → remainder of divison

Strings

To use methods in strings just type val name and . method name

If condition

- if can be used as an expression

- eg→

```
fun main() {
val x = 2
val y = if(x == 2) 2 else 3
println (y)
}
```

- The value is set to 2 if the if condition passes and 3 if it fails

Null values

There are nullable and non nullable types

eg → val x: Int? = null

- A ? is added if the type is not equal to null
- RESEARCH ABOUT NULL VALUES
- In the above example we can see how ? is added after Int

To typecast

- var_name.toInt()
- var_name.toString()

Lists

use keyword → listOf<datatype>(values)

[ ] → index of the lists

The above keyword is used for immutable lists

For mutable it is,

mutableListOf(values)

no need to specify datatype

list.add(index,value)

Loops

The looping system is traditional with while

and for has a similar syntax as python

but for specifying range or number of iterations in for loop

- for (i in 1 .. 100) {}

When Expression

```
Syntax → when ( x) {
    in 1..2 → println(”Hey”)
    in 3..10 → println(”yeh”)
    else →{
        println(”……“)
        }
    }
```

Similar to if condition

depends on one variable

Function

keyword fun

ex → fun f_name() : return_type{ }

—> TO MAKE ANY DATATYPE OF A VARIABLE NOT OPTIONAL USE

name: datatype

Crlt + P → to check what parameters to send

Default parameters are possible

```
fun Int.isOdd(): Boolean {
    return this % 2 == 1
}
```

Class

instance is created in the main file

```
class Animal (
    val name: String
    }
```

- the Name we put in the parenthesis is a constructor which is necessary to enter when calling it

```
    {
        init(
            // proceeds to compile when instance is created
            println(" so on")
        )
    }
```