→ Type_Conversion:-

*) :-

Type Conversion (Type casting) refers to changing the entity of one data type variable into another data type.

*) Java supports implicit type conversion from smaller to larger data types. An Integer value can be assigned to the long data type

*) But Kotlin doesn't support implicit type conversion.

Integer cannot assigned to long data type in Kotlin

*) In Kotlin, the help functions are used to explicit convert one data type to another data type.

Q) What are help functions?

A) The help functions are - i) toByte(), ii) toShort(), 3) toInt(), 4) toLong(), 5) toFloat(), 6) toDouble(), 7) toChar()

→ So total 7 help function.

Now: If I have string input with the help of help functions Can I convert to the Integer data type?

A) Exception error (NumberFormatException)

(*) Note: There is no helper function available to convert into boolean type.

→ Program: uploaded the Kotlin file with name type_conversion.kt Please check out.

*) Kotlin Expression, Statement, & Block:-

1) Expression:- It basically contains variables, operators, method calls etc that produce a single value.

Kotlin Expression is a building blocks of a program that are usually created to produce new value.

→ Expression gives us a single value and that value is sometimes assigned to the variable.

→ Expression can contain another expression.

*) var a = 100 (variable declaration, & we should not assume this as an expression)

*) Assigning a value is ~~an~~ not an expression (b=15).

*) A class declaration is not an expression (class XYZ {...})

Note:- In Kotlin, function return a value atleast Unit, so every function is an expression.

Q):-

→ Program:- Identify & give me reasons what are expressions & why they are Expression:-

*) Note:- program uses concept of functions, you are not still aware of function concept it is okay.

→ fun sumof(a:Int, b:Int):Int { return a+b; } → is this a function it is, ~~an~~ not Expression why? function returns a value atleast unit: so every function is an Expression.

Not a single expression function

fun main(args: Array<String>) { val a=10; → Not Expression, because variable declaration is not an Expression.

Expression or not?

val b=5;

var = sumof(a,b) → Expression (refer 1st line of the page).

var mul = a*b; → a*b is an expression, that value is assigned to variable.

}

→ clarification:-

*) function calls are expressions

*) function definition can be expression, particularly in the case of single-expression functions.

Q) What exactly is single-expression function?

A):- In Kotlin single-expression function is a function that consists of single expression to be evaluated.

Ex:- fun add(a:Int, b:Int):Int = a+b

→ single expression function can omit curly braces { } and the 'return keyword'. The return type can explicitely declared

*):- Note:- In Kotlin "main function does not produce a value" it simply initiates the execution of the program. Therefore, it is not classified as an expression in Kotlin.

*):- In 'java' 'if' is an statement, but in Kotlin 'if' is an expression it is called an expression because it compares the value of 'a' and 'b' and returns the maximum value. Therefore, In Kotlin there is no-ternary operator (a>b)?a:b because it is replaced by 'if' expression.

→ var max = if(a>b) a else b.

→ A:- if (condition) condition is satisfied else condition is not satisfied.

→ Kotlin statement:-

*) A statement is a syntactic unit of programming that expresses some actions to be carried out. A program is formed by sequence of one or more statements.

→ Multiple Statements:

*) :- In the context when you write more than one statement in a single line.

Ex:- println("Puneeth"); println("Madhavi"); ——) Multiple statement

*) :- Kotlin block:

1) :- A block is a section of software code enclosed with curly braces ({ ... })

2) Block contains so many statements.

3) Block can contain one or more blocks nested within it.

4) Every function has its own block & main function also contains a block.

→ When we talk about block, scope of variables comes into picture:-

*) Note:- The variables declared at the head of the block are visible throughout the block and any nested blocks, unless a variable with the same name declared at the head in the inner block. When a new declaration is effective through out the inner block the outer declaration becomes effective ~~against~~ again at the end of the inner blocks.

Conclusion:- Variables haves nested scopes —— True

(*) CONTROL FLOW

=)