CS 131
PROGRAMMING LANGUAGES
(WEEK 9)

UCLA WINTER 2019

TA: SHRUTI SHARAN

DISCUSSION SECTION: 1D

ADMINISTRATIVE INTRODUCTION

TA: Shruti Sharan

Email: shruti5596@g.ucla.edu
(Will reply by EOD)

Office Hours: Mondays 1.30PM – 3.30PM
Location: Eng. VI 3rd Floor

Discussion Section: Friday 4.00-5.50PM
Location: 2214 Public Affairs

ANNOUNCEMENTS

- Course feedback is open
  - Fill by next week. Deadline Saturday 8.00 AM (03/16/2019)
  - What should be changed for next quarter?
  - Anything you wish would have been covered in the discussion sections?
  - What would have been helpful when doing the homeworks?
- Project deadline has been shifted to Tuesday 11:55PM (03/12/19)
- Homework 6 is due next Friday 11.55 PM (03/15/19)
  - No late submissions after Friday due to finals week
  - Only a short coding question and a report. (Yaay?)

FINAL EXAM REVIEW

- What would you like to see covered?
- What are you most and least comfortable with?
- What sort of examples would you prefer?

TODAY'S AGENDA

MACHINE LEARNING

TENSORFLOW

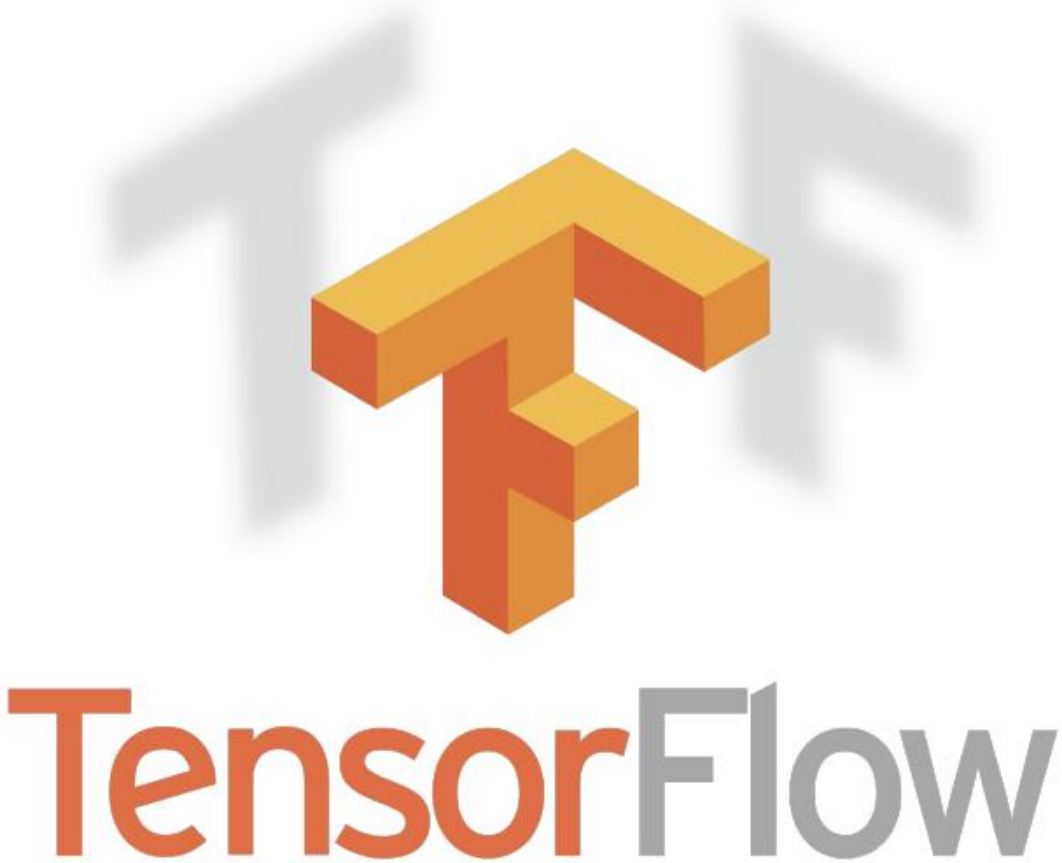
LANGUAGE BINDINGS

KOTLIN

#HW6 - OVERVIEW

- TensorFlow – Machine Learning
  - Python Binding
- Small models
  - Bottleneck in performance on Python code
- Report on possible alternatives
  - Must support event-driven servers
  - Options: Java, OCaml, Kotlin
- Evaluate based on:
  - “ease of use, flexibility, generality, performance, reliability”

Machine Learning & TensorFlow

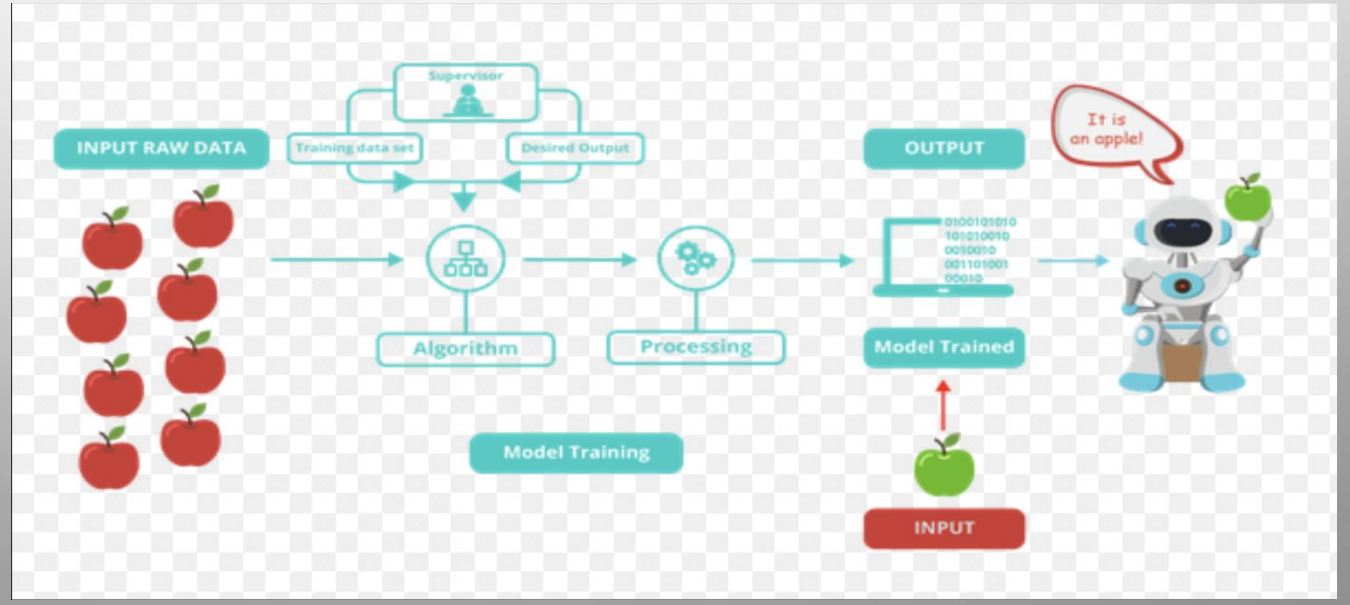


MACHINE LEARNING

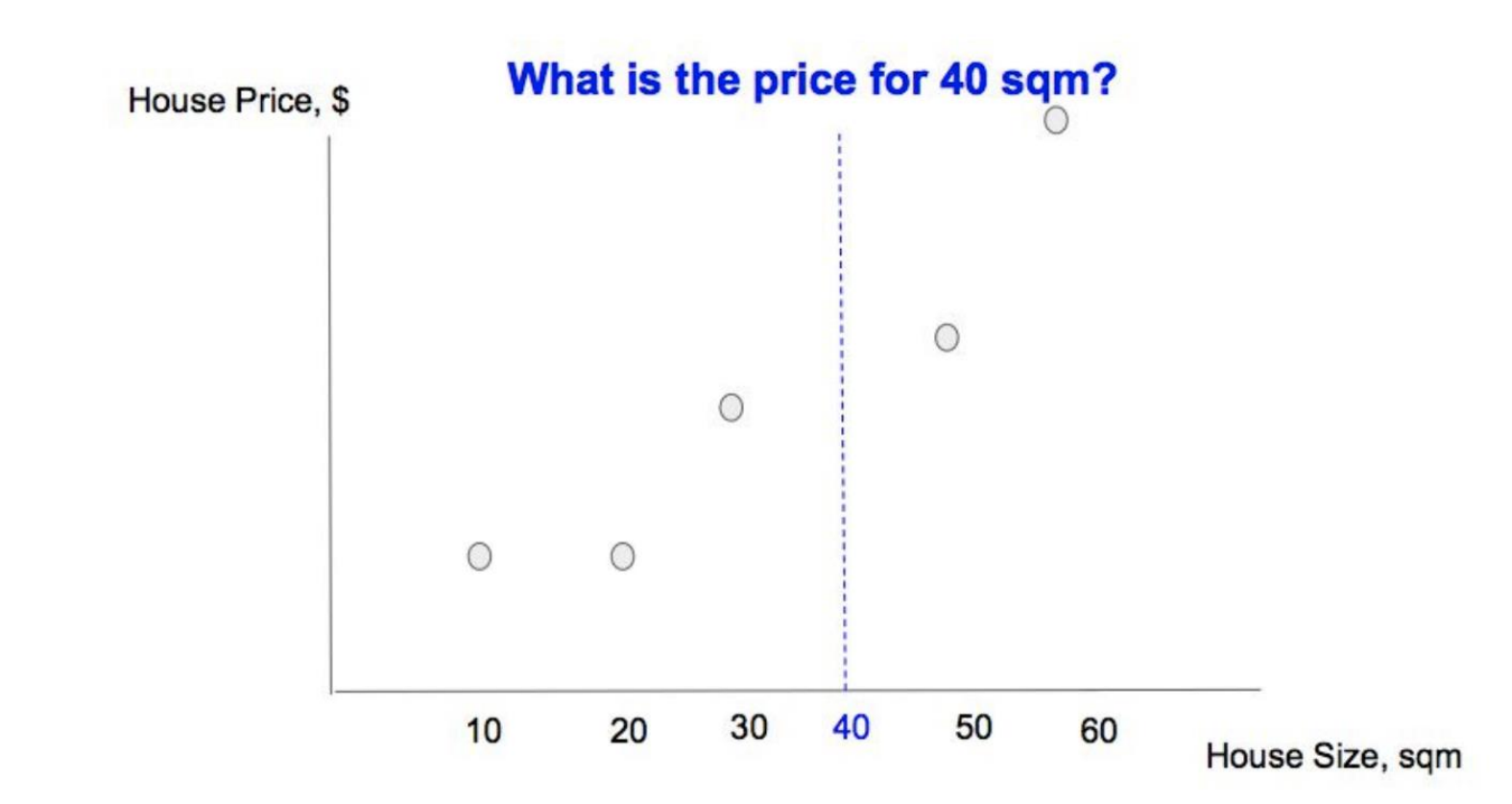
- Use of algorithms and mathematical models to solve a problem
- Goal is for the system to improve its performance as it “learns”
- Compare with your typical programs, where you explicitly define what should happen in each situation
- Three main categories:
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning

MACHINE LEARNING : SUPERVISED LEARNING

- System tries to learn from examples given to it
- For example, a classifier can learn how an apple looks like and tell an apple from different fruits.



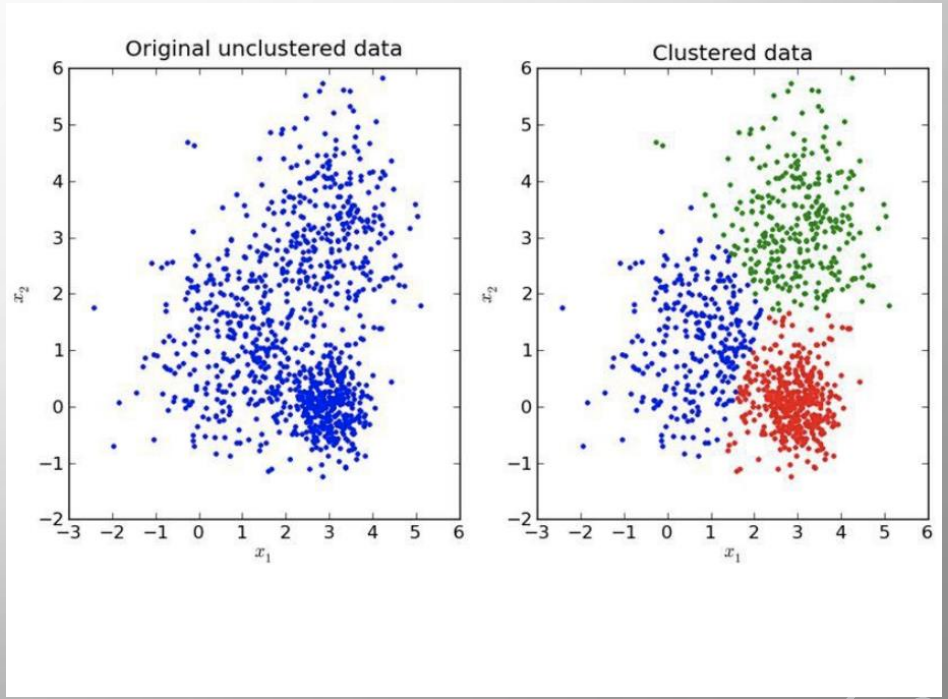
SUPERVISED LEARNING



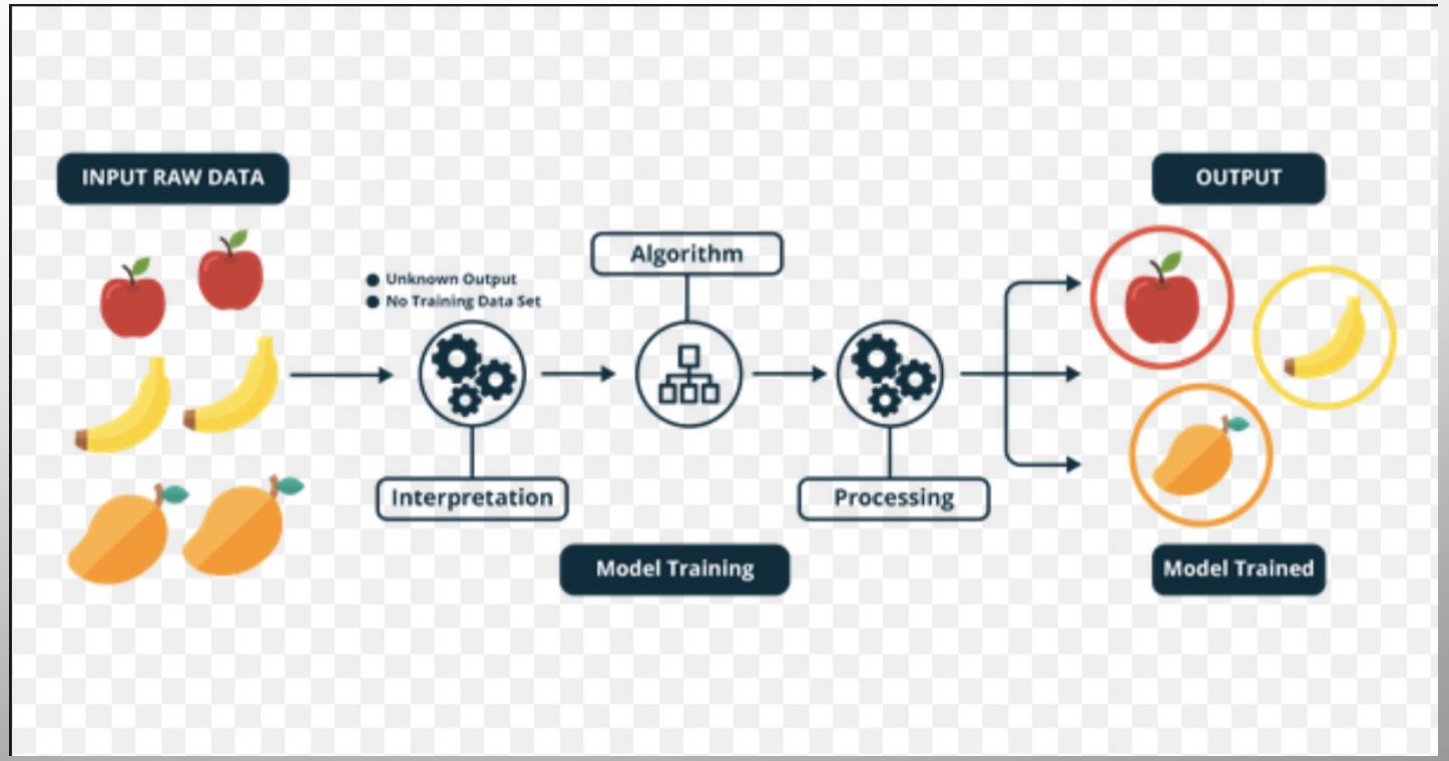
SUPERVISED LEARNING

MACHINE LEARNING - UNSUPERVISED LEARNING

- System tries to find structure from data
- For example, which users are likely to enjoy a specific movie on Netflix?
- Find groups of users and see which groups have lots of people enjoying that movie
- Is a credit card transaction typical for that user, or possibly a fraud?
- Detect outliers

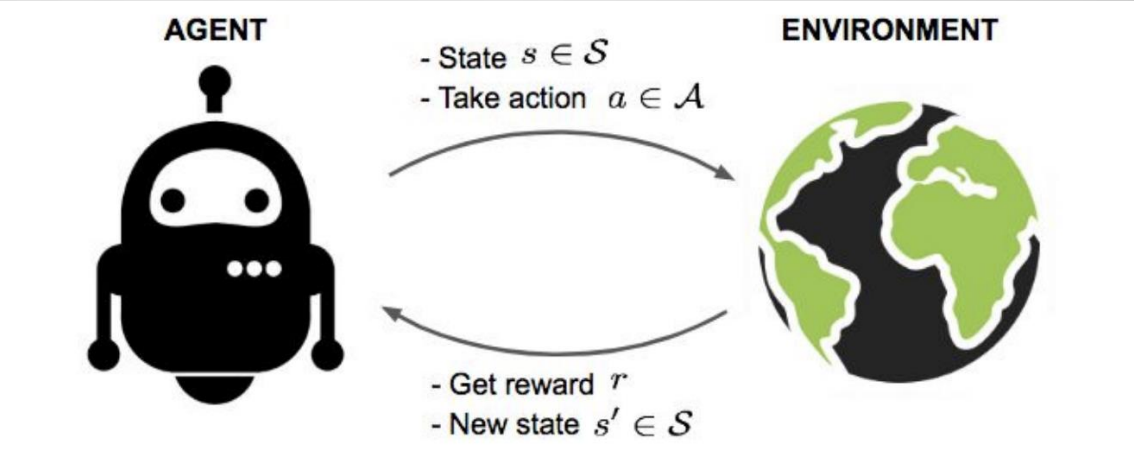


UNSUPERVISED LEARNING

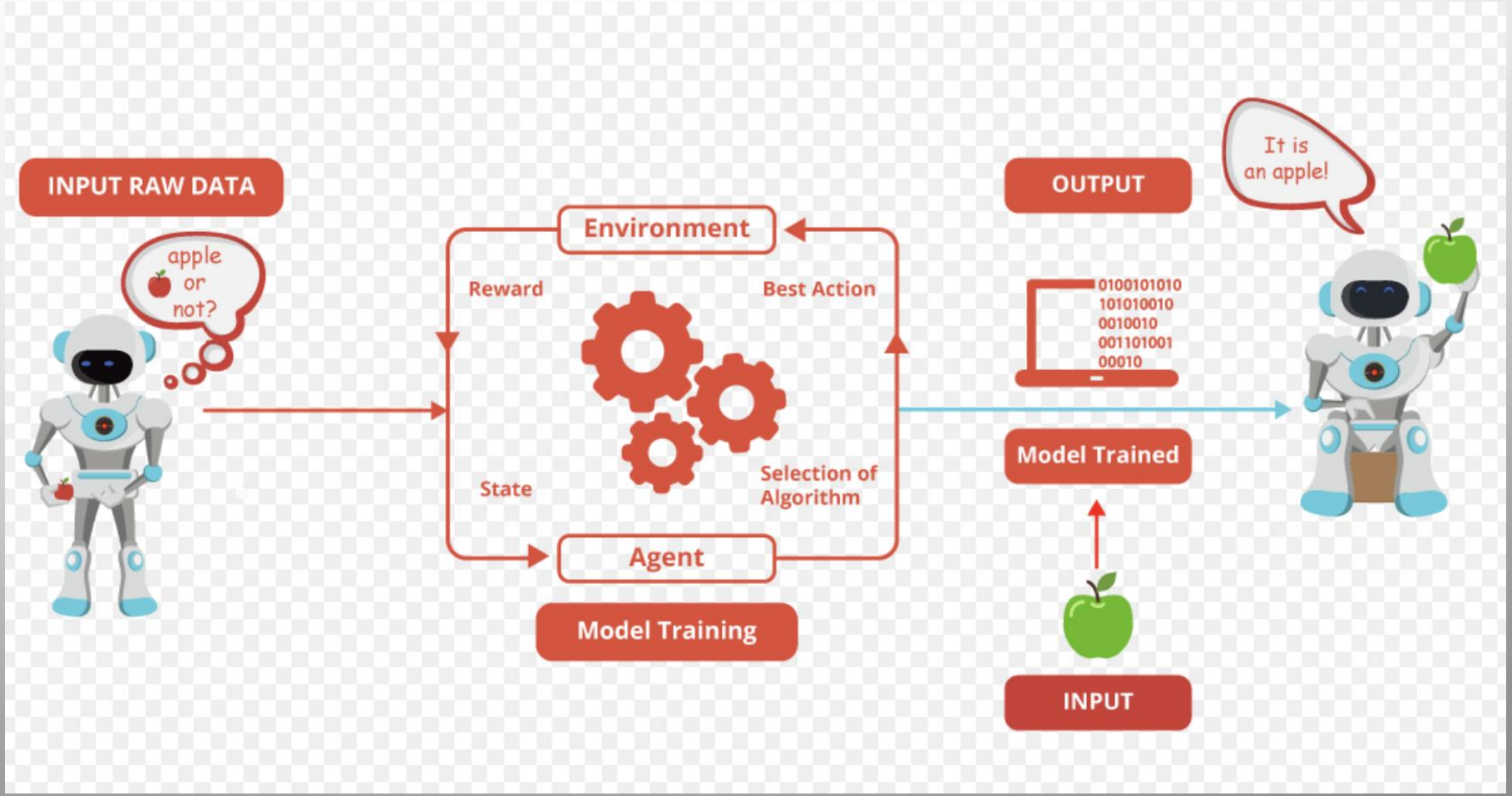


MACHINE LEARNING - REINFORCEMENT LEARNING

- Try to perform different actions and see what the outcome is
- Learn from the successes/mistakes
- For example, a program that can play computer games without having rules explicitly coded into it

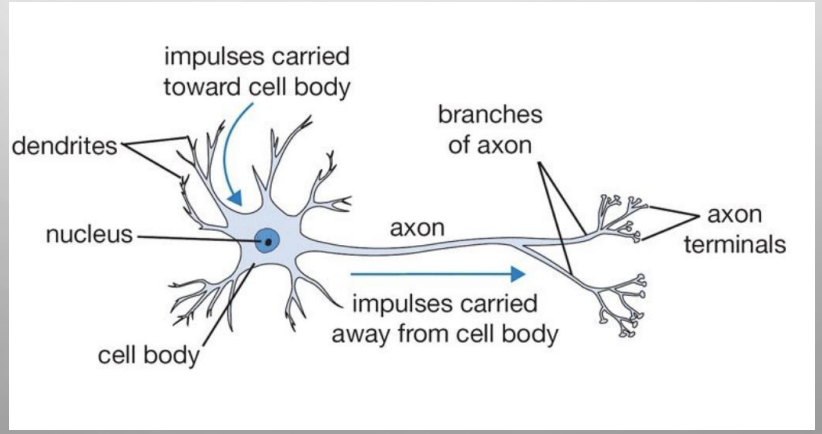


REINFORCEMENT LEARNING



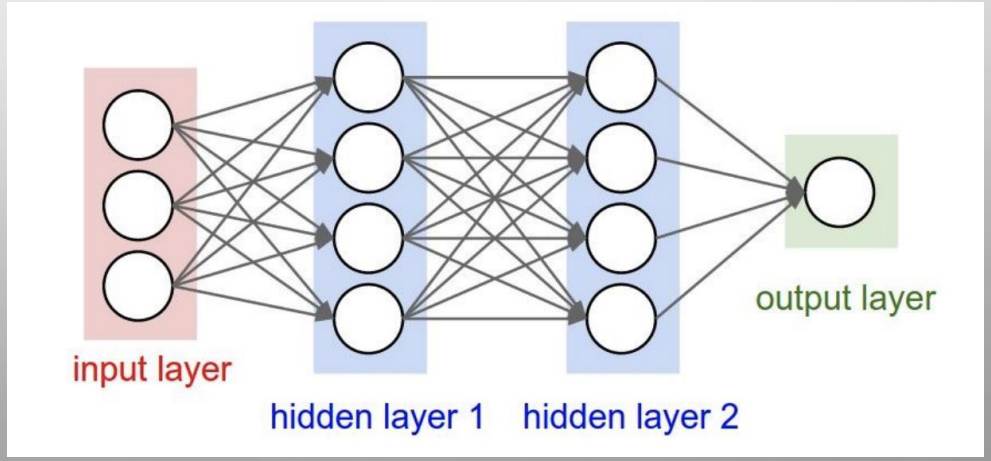
NEURAL NETWORKS

- Powerful machine learning approach
- has been shown to have great performance on tasks such as image recognition or signal processing
- Challenge: computationally heavy, difficult to reason about (Black box)



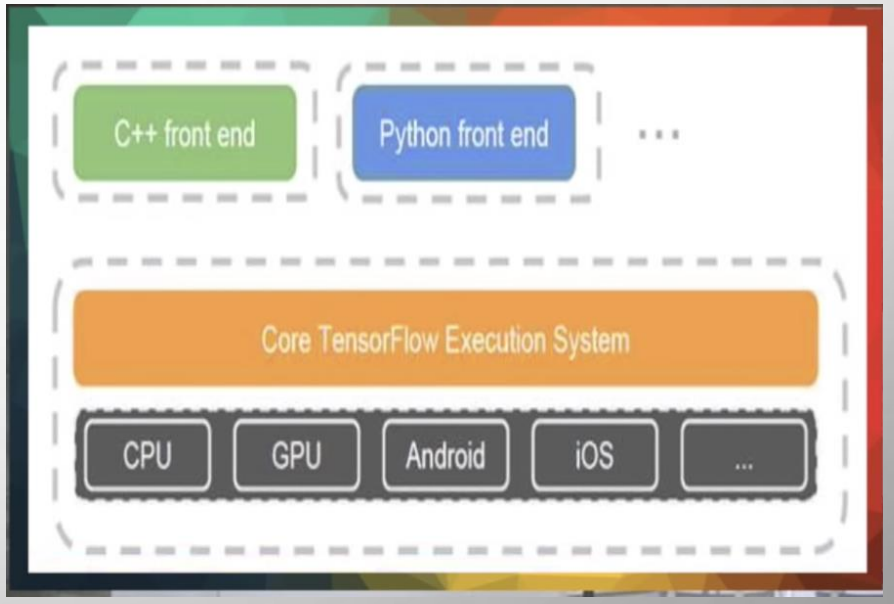
NEURAL NETWORKS

- Neural networks consist of a large number of neurons, each performing a simple mathematical operation on its input values
- If interested, consider taking ECE239AS - Neural Networks and Deep Learning.



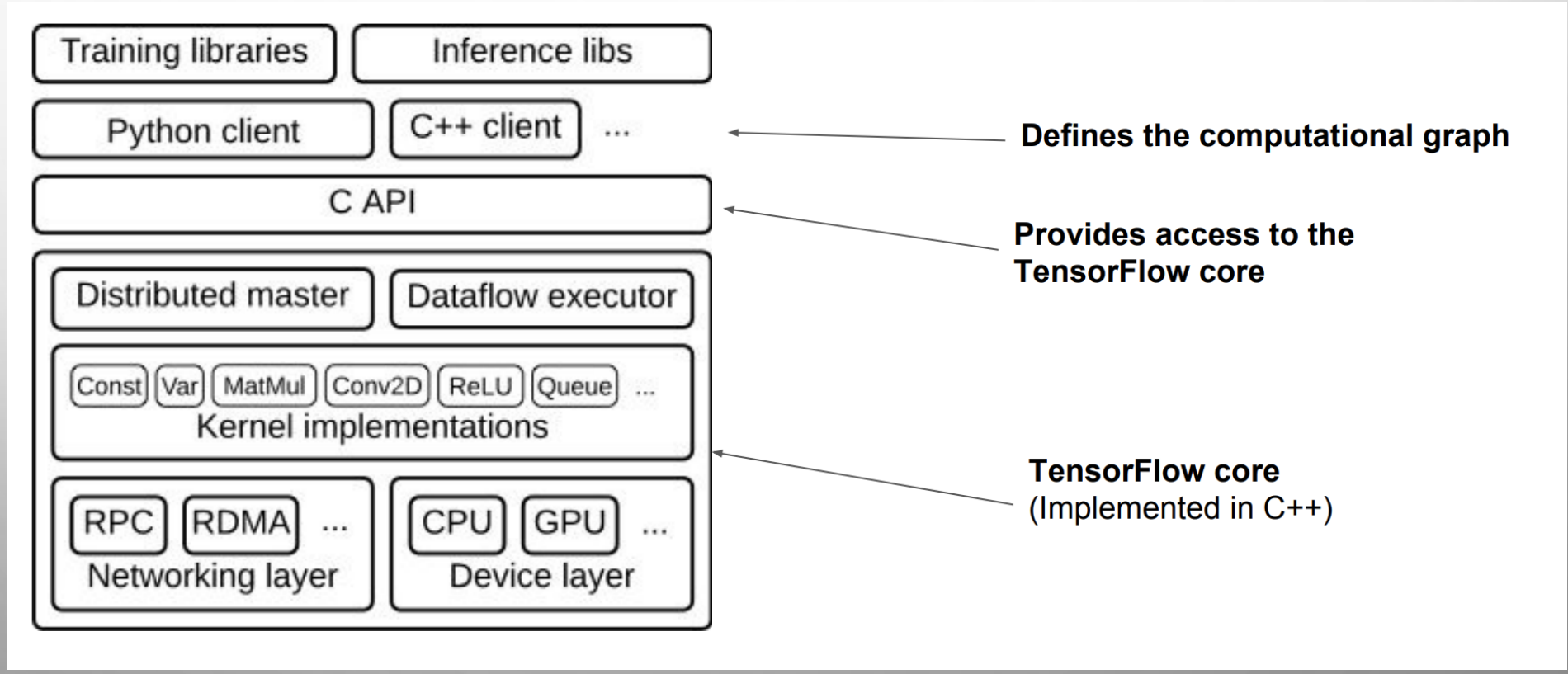
TENSORFLOW BASICS

- Developed by Google
- Implemented in C++
  - Core features made here
  - Fleshing out C API for future bindings
- Used for ML model creation and testing
- Easy to define neural networks, TensorFlow takes care of running it efficiently on multiple CPUs or GPUs.



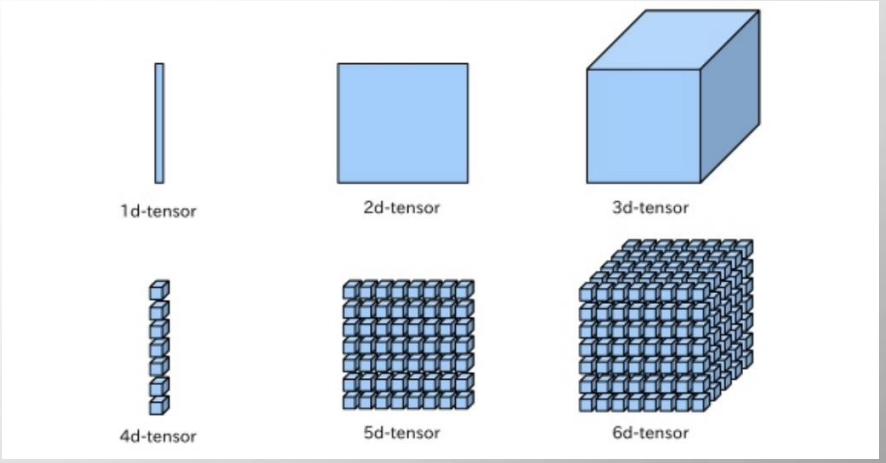
TENSORFLOW ARCHITECTURE

- (Documentation)



TENSORFLOW BASICS

- **Tensors**
  - Basic unit of data
  - Essentially a multidimensional array
  - Edges on the TensorFlow Graph



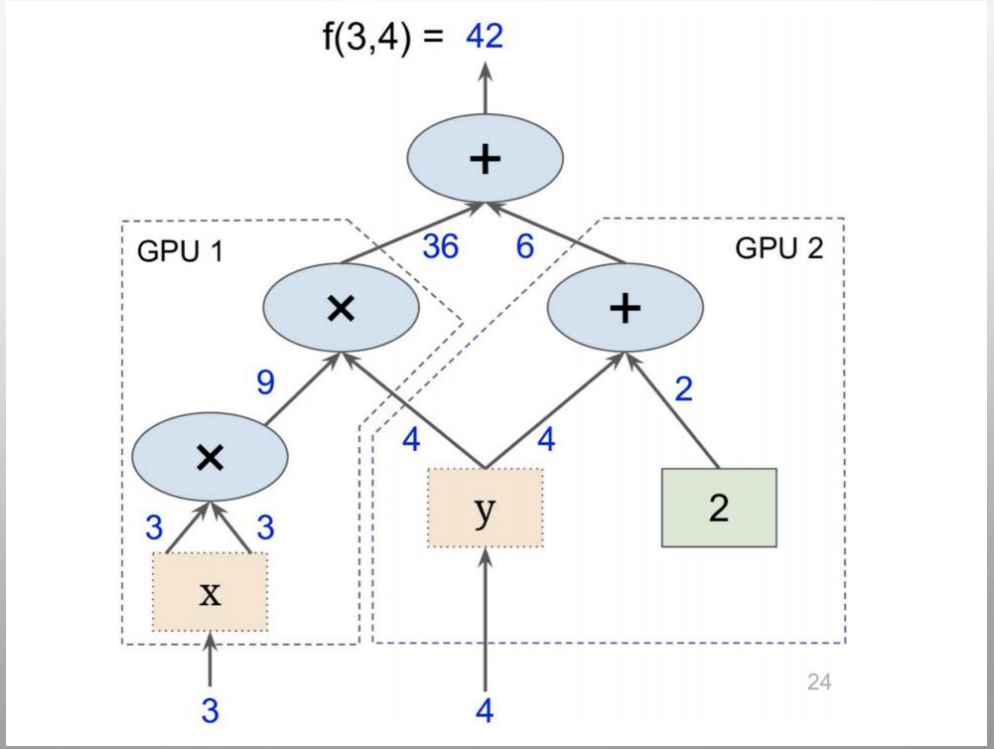

```
3. # a rank 0 tensor; a scalar with shape [],
[1., 2., 3.] # a rank 1 tensor; a vector with shape [3]
[[1., 2., 3.], [4., 5., 6.]] # a rank 2 tensor; a matrix with shape [2, 3]
[[[1., 2., 3.]], [[7., 8., 9.]]] # a rank 3 tensor with shape [2, 1, 3]
```

TENSORFLOW

- Tensorflow separates definition of the model from the computation.
- The model definition is in the form of a graph
- **Operations**
  - Calculations that consume and produce tensors
  - Nodes on the TensorFlow Graph
  - TensorFlow Basics Tutorial

DATAFLOW GRAPHS

- Important because subgraphs can be used to optimize computations



TENSORFLOW: HOW IT WORKS.

- Google Colab Tensorflow Demo: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_classification.ipynb
- Code for the above implementation can be found here.

LANGUAGE BINDING

- What is language binding?
- Why is it useful?
- Why not just rewrite for a given language?

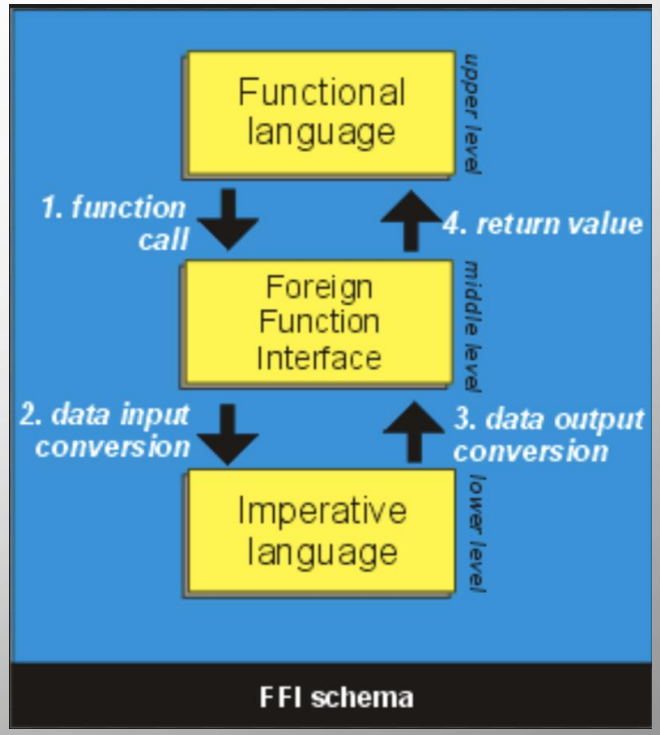
- Most libraries are built in system programming languages like C or C++
- To use these libraries in other languages we have to create a binding.
- Why?
  - Code reuse
  - Somethings are better done at a lower-level

LANGUAGE BINDING - METHODS

- Foreign Function Interfaces (FFI)
- Inter-Process Communication
- Interface Compilation
  - SWIG

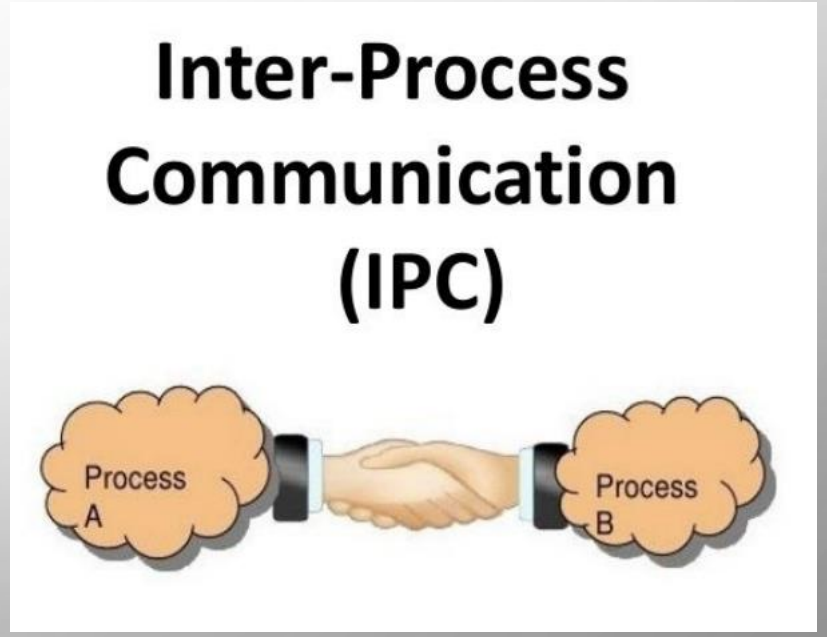
FOREIGN FUNCTION INTERFACES (FFI)

- Main Idea:
  - Merge semantics and calling conventions between a host language and a guest language
  - Something written in one language can call a routine written in another
  - Directly calls functionality to be used at runtime



INTER-PROCESS COMMUNICATION

- Main Idea:
  - Two separate processes that communicate with each other
  - Use an intermediary to form and pass data between processes
  - Think of it like a client / server system
    - One process makes a request, the other process fills that request



INTERFACE COMPILATION

- Main Idea:
  - Connect existing libraries with another language by generating wrapper code to connect the two
  - Generate and compile code that calls library while conforming to the given new language conventions
  - Example - SWIG
    - http://www.swig.org/exec.html
    - Connects programs written in C and C++ with scripting languages such as Perl, Python, Ruby, and Tcl.

TENSORFLOW BINDINGS

- Python API has been the main focus of the TensorFlow team, so it has all the latest features
- Bindings exist also for JavaScript, C++, Java, Go, Swift, C#, Haskell, Julia, Ruby, Rust, OCaml, and Scala
- These bindings are not as mature as the Python bindings
- Some features might not be available in all of them , some of these are provided by TensorFlow team, some not.

TENSORFLOW BINDING

- Core Features Required of a Binding:
  - Run a predefined graph
  - Graph construction
  - Gradients (automatic differentiation)
  - Functions
  - Control Flow
  - Neural Network Library

What does our binding need to keep track of to implement these features?

TENSORFLOW IN OTHER LANGUAGES

- You can use TensorFlow C API to add support to any language.
- Tensorflow Bindings in other languages: Documentation
- Tensorflow in Java:
  - https://github.com/tensorflow/tensorflow/tree/master/tensorflow/java
- Tensorflow in Ocaml:
  - https://github.com/LaurentMazare/tensorflow-ocaml/
- Tensorflow in Kotlin:
  - https://juliuskunze.com/tensorflow-in-kotlin-native.html

KOTLIN

KOTLIN FEATURES

- Statically typed
- JVM Based language (Like Java, Scala, JRuby.. Etc)
- Object Oriented Language
- Supports functional programming with lambda and higher order functions.
- Interoperability (Java + JavaScript)
- Avoids NullPointerException

Why Kotlin?

Concise

Drastically reduce the amount of boilerplate code.

See example

Safe

Avoid entire classes of errors such as null pointer exceptions.

See example

Interoperable

Leverage existing libraries for the JVM, Android, and the browser.

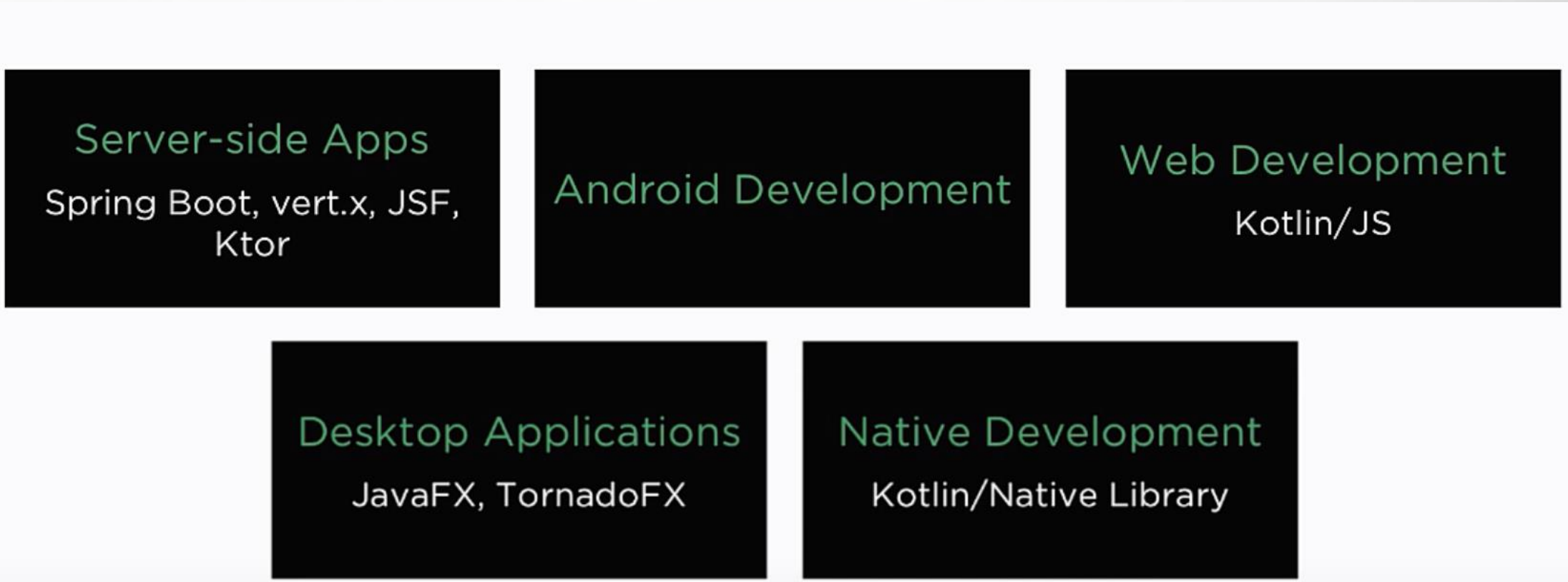
See example

Tool-friendly

Choose any Java IDE or build from the command line.

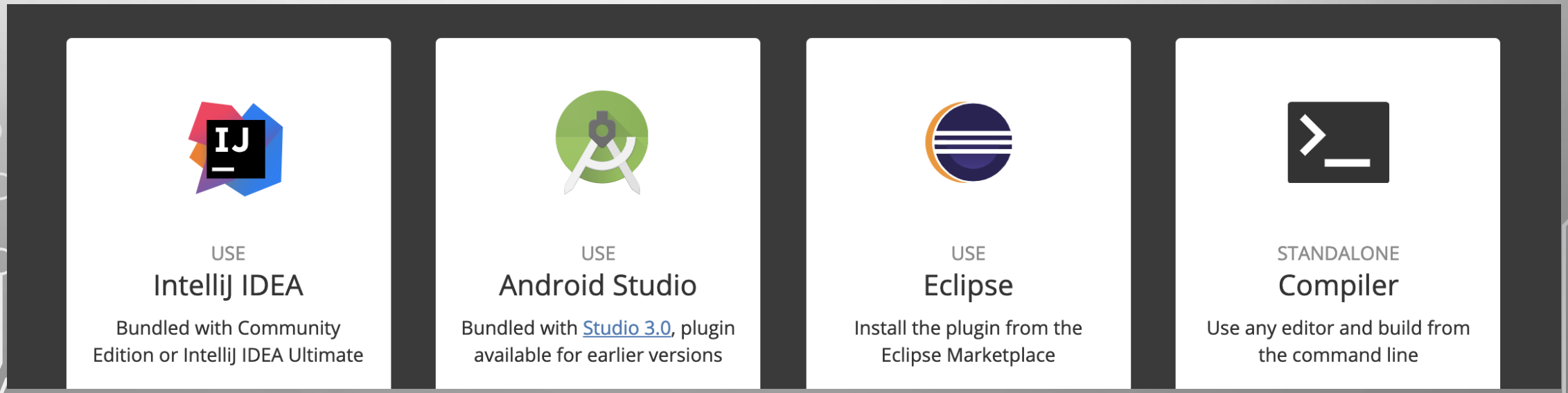
See example

TYPES OF APPLICATIONS OF KOTLIN



KOTLIN INSTALLATION

- Runs on JDK (Just like JAVA)
- Use any of the IDEs (ItelliJ IDEA, Eclipse etc)
- Install it and use your favorite editor to build it and run from terminal.



JAVA VS KOTLIN

```java
class HelloWorld
{
    public static void main(String args[])
    {
    System.out.println("Hello World");
    }
}
```

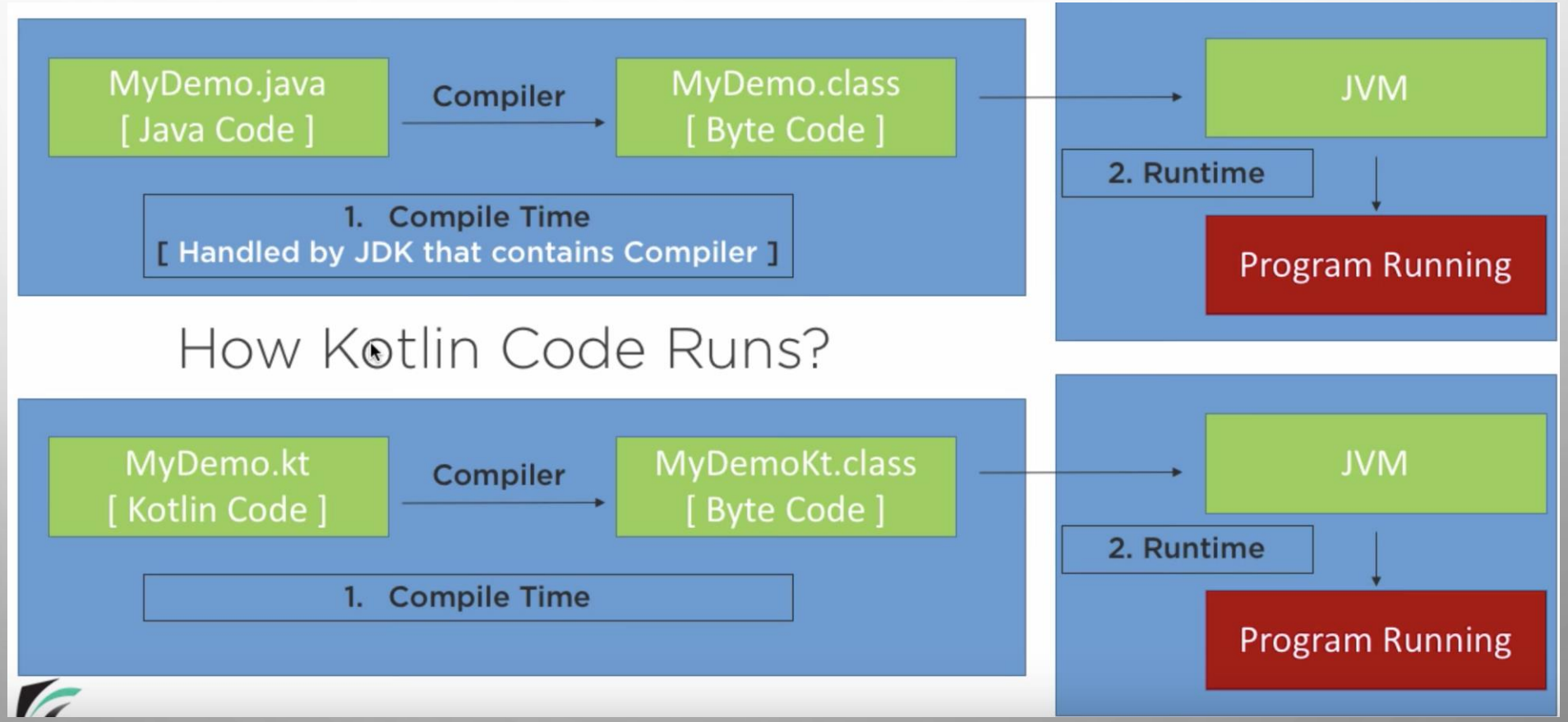
```
[Shruti@Kotlin$cat hello.kt
fun main(args: Array<String>)
{
  println("Hello World")
}
```

GETTING STARTED

- Write your program in file with .kt extension.
- Compile your program using kotlinc (Kotlin Compiler) command.
- This creates a Kotlin class file.
- Run the class file using kotlin command.

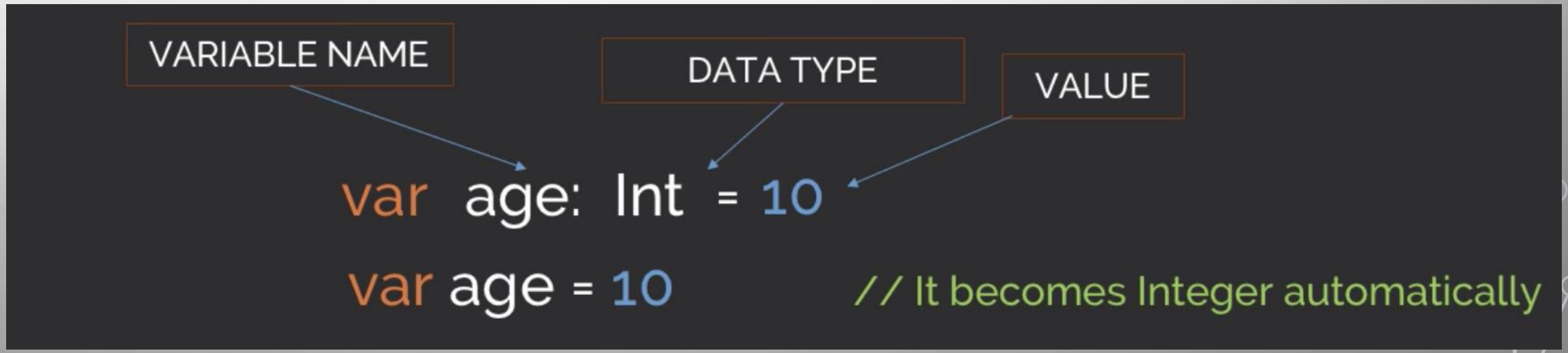
```
[Shruti@Kotlin$cat hello.kt
fun main(args: Array<String>)
{
  println("Hello World")
}
```

JAVA VS KOTLIN



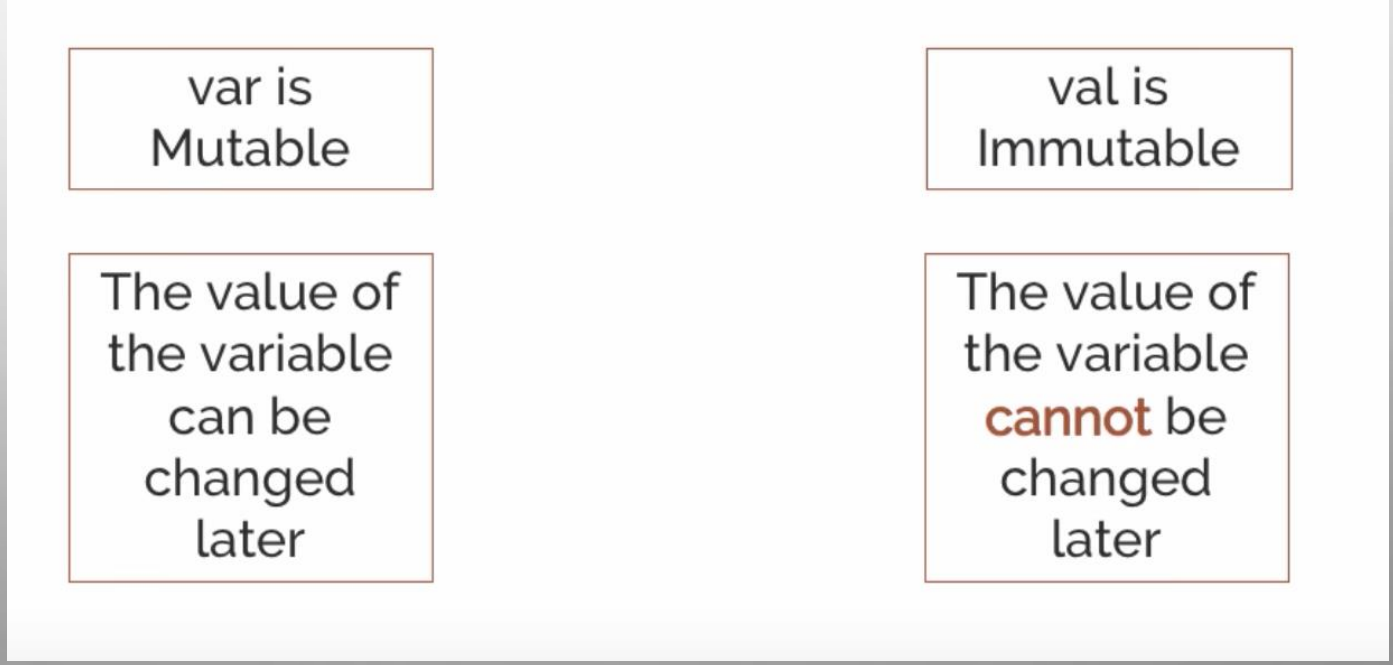
VARIABLES IN KOTLIN

- Data type is inferred, need not be specified explicitly.
- Type required when no initializer is provided



VARIABLES AND VALUES

- Read-only local variables are defined using the keyword val
- Variables that can be reassigned use the var keyword:

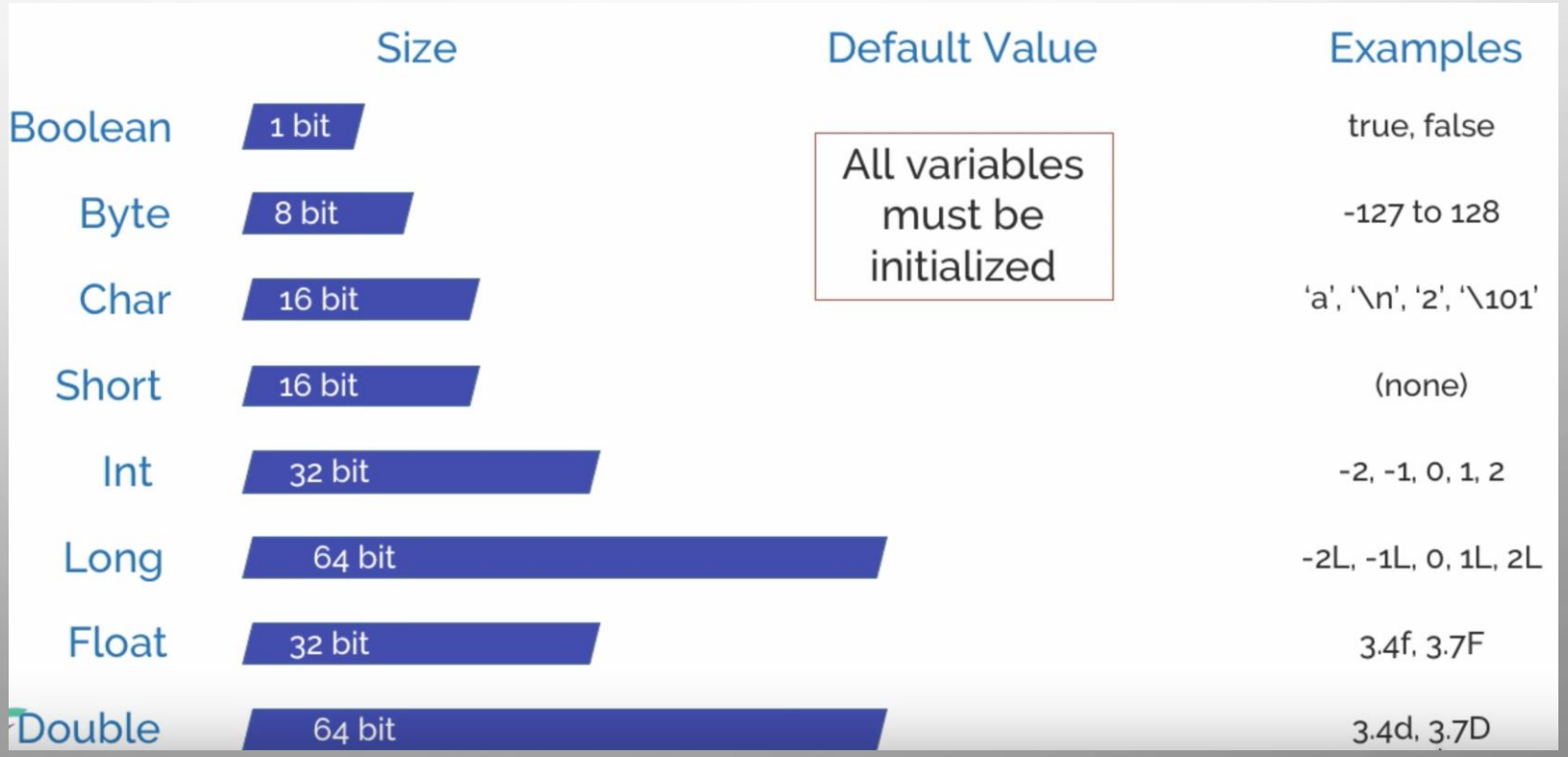


VARIABLES AND VALUES

```
fun main(args: Array<String>) {

    var myNumber = 10          // Int
    var myDecimal = 1.0        // Float

    var myString: String       // String
    myString = "Hello World"
    myString = "Another World"

    val myAnotherStr = "My Constant String Value"

    print(myNumber)
}
```

DATATYPES: ALL ARE OBJECTS

- There are no primitive datatypes in Kotlin.



RANGES

- Range expressions are formed with rangeTo functions that have the operator form .. (double dot) which is complemented by in and !in.

```
fun main(args: Array<String>) {

    // Ranges

    val r1 = 1..5
    // This range contains the number 1, 2, 3, 4, 5

    val r2 = 5 downTo 1
    // This range contains the number 5, 4, 3, 2, 1

    val r3 = 5 downTo 1 step 2
    // This range contains the number 5, 3, 1

    var r4 = 'a'..'z'
    // This range contains the values from "a", "b", "c" .... "z"

    var isPresent = 'c' in r4

}
```

```
var countDown = 10.downTo(1)
// This range contains the number 10, 9, 8, 7, 6.....1

var moveUp = 1.rangeTo(10)
// This range contains the number 1,2,3,4,5,6.....10
```

IF EXPRESSIONS

```
fun main(args: Array<String>) {

    // IF as Expression

    val a = 2
    val b = 5

    var maxValue:Int

    if (a > b)
        maxValue = a
    else
        maxValue = b

    println(maxValue)
}
```

```
fun main(args: Array<String>) {

    // IF as Expression

    val a = 2
    val b = 5

    var maxValue:Int = if (a > b)
                           a
                       else
                           b

    println(maxValue)
}
```

- If as an expression returns a value that van be assigned to a variable.

WHEN EXPRESSIONS

- Used instead of Switch Case in C++, Java
- No break statements required
- Default is implemented using else keyword.
- You can combine multiple conditions into one as well.

```
when (x) {

    0, 1 -> println("x is 0 OR 1")
    2 -> {
        println("x is 2")
    }
```

```
fun main(args: Array<String>) {

    // WHEN as Expression

    val x = 2

    when (x) {

        1 -> println("x is 1")
        2 -> println("x is 2")
        else -> println("x value is unknown")
    }

}
```

WHEN AS AN EXPRESSION

- When is used to evaluate some expression and return a value.
- The last executed statement is returned from each block.
  - The other expressions are redundant code.

```
var str:String = when (x) {
    1 -> "x is 1"
    2 -> "x is 2"
    else -> {
        "x value is unknown"
    }
}
```

```
var str:String = when (x) {
    1 -> "x is 1"
    2 -> "x is 2"
    else -> {
        "x value is unknown"
        "x is an alien"
    }
}
```

ITERATORS

```
println("Hello");
println("Hello");
println("Hello");
println("Hello");
```

Output:

Hello

Hello

Hello

Hello

Loop Structure:

- Counter Variable
- Condition Check
- Increment/Decrement the counter

FOR LOOP:

```
for ( i in 1. .4 ) {
        println("Hello");
}
```

WHILE LOOP:

```
while (i < 5) {
        println("Hello")
        i++
}
```

DO WHILE LOOP:

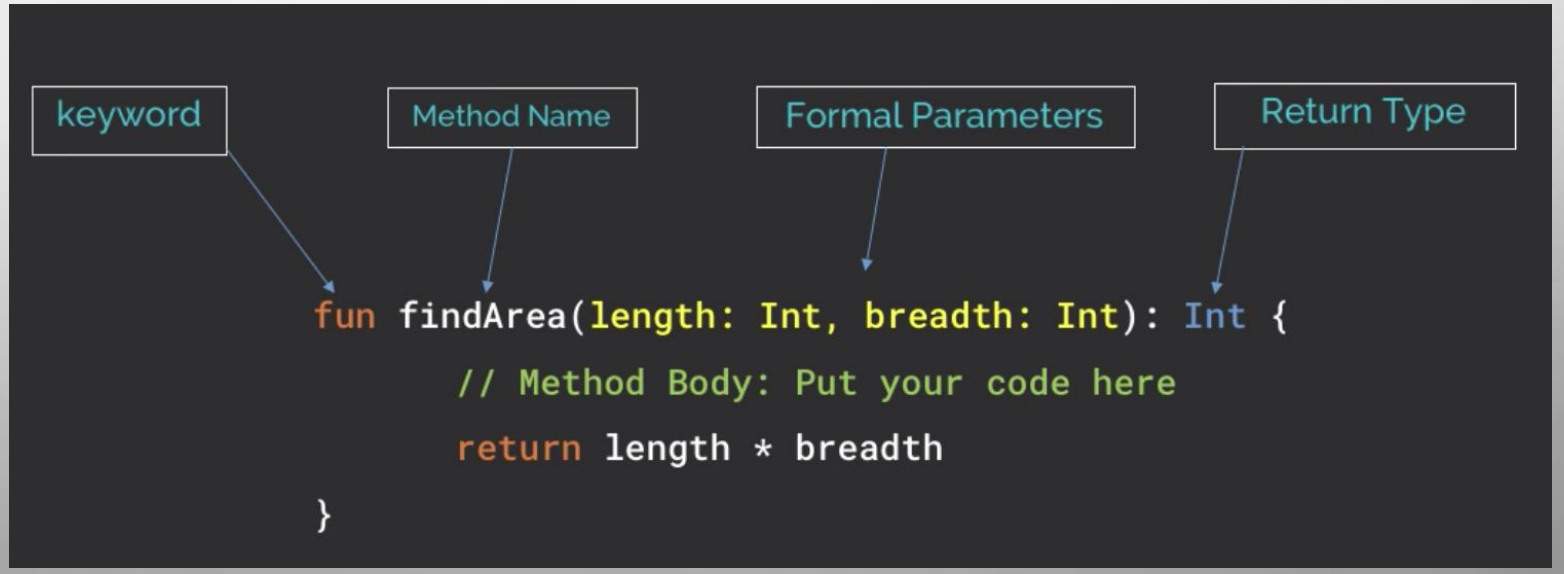
```
do {
        println("Hello")
        i++
} while ( i < 5 )
```

LOOP CONTROL STATEMENTS

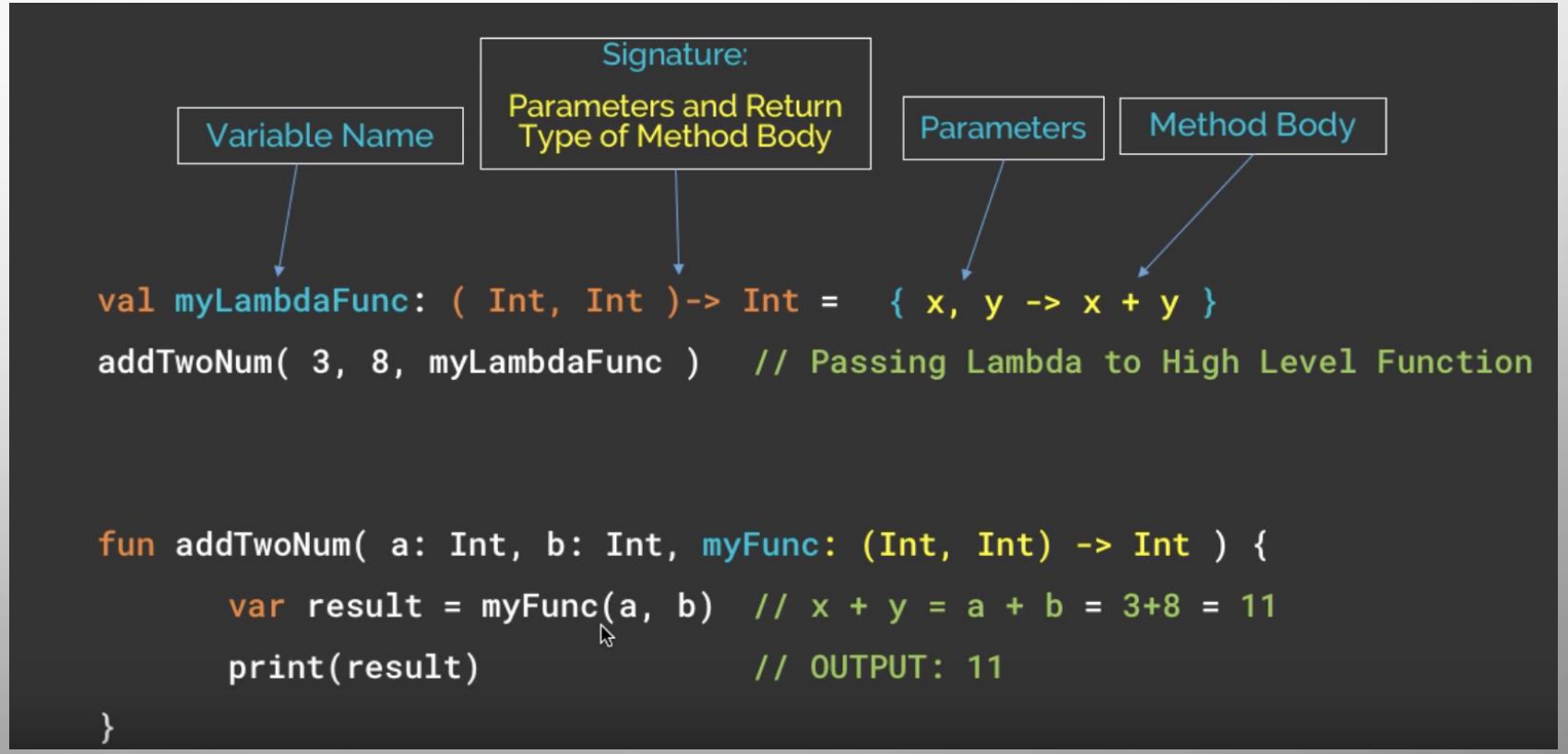
Same as Java

METHOD STRUCTURE

- Functions in Kotlin are declared using the fun keyword:
- If a method returns no value (void), the keyword Unit is used. (Like Ocaml)



LAMBDA FUNCTIONS



COLLECTIONS

Array: Mutable but has Fixed Size

Collections

- → Immutable Collections: Read Only Operations
  - Immutable List: listOf
  - Immutable Map: mapOf
  - Immutable Set: setOf

- → Mutable Collections: Read and Write Both
  - Mutable List: ArrayList, arrayListOf, mutableListOf
  - Mutable Map: HashMap, hashMapOf, mutableMapOf
  - Mutable Set: mutableSetOf, hashSetOf

LISTS

```
fun main(args: Array<String>)
{
  var list = listOf<String> ("Apple", "Oranges", "Bananas")
             //Immutable fixed Size List

   for(element in list)
    {
       println(element)
    }

    for(index in 0..list.size -1)
     {
       println(list[index])    //list.get(index)
     }
}
```

```
[Shruti@Kotlin$kotlin ListKt
Apple
Oranges
Bananas
Apple
Oranges
Bananas
```

NEW FEATURE : NULL SAFETY

- Does not support ‘null’ to avoid NullPointerException
- Makes programs more robust and error safe.
- A reference must be explicitly marked as nullable when null value is possible.

NULL SAFETY OPERATORS

```
// WAP to find out length of name
val name: String? = "Steve"

// 1. Safe Call ( ?. )
    // Returns the length if 'name' is not null else returns NULL
    // Use it if you don't mind getting NULL value
println("The length of name is ${name?.length}")
```

```
// 2. Safe Call with let ( ?.let )
    // It executes the block ONLY IF name is NOT NULL
name?.let { it: String
    println("The length of name is ${name.length}")
}
```

```
// 3. Elvis-operator ( ?: )
// When we have nullable reference 'name', we can say "is name is not null", use it,
// otherwise use some non-null value"
val len = if (name != null)
            name.length
        else
            -1

val length = name?.length ?: -1
print("The length of name is ${length}")
```

```
// 4. Non-null assertion operator ( !! )
    // Use it when you are sure the value is NOT NULL
    // Throws NullPointerException if the value is found to be NULL

print("The length of name is ${name!!.length}")
```

#HW6: KOTLIN

- Write and test a simple method everyNth that accepts a list L and a positive integer N and returns a list containing every Nth element of L, starting with the (N−1)st element, then the (2N−1)st element, and so on until L's elements are exhausted so that the returned list's size is ⌊L.size / N⌋ (where ⌊*x*⌋ is the floor of *x*).
  - Eg. List=1,2,3,4,5,6 ; N=3
  - Output: 3 (N-1=2nd position) , 6 (2N-1=5th position)
- The method should accept a list of any type, should return a list of the same type, and should consume O(N) time and space.
- The returned value should be immutable; any later changes to the input list should not affect the output list.

#HW6: REPORT

- Tips on getting started:
- Check TensorFlow bindings for Python, Java, OCaml, Kotlin
  - Do they provide the same features?
- Would you use the API for a real application?
- How would you build the server using these languages?
- Are some languages better for it than others?
- What libraries are available for implementing the server?
- Are there differences in performance between languages?
- Any other differences that should be taken into account?
- No coding required; write a 3 page report that can be understood by people who are not experts in any of the given languages but understand software

RESOURCES

- Useful Reading:
- https://www.oreilly.com/learning/hello-tensorflow
- https://www.tensorflow.org/tutorials
- https://www.tensorflow.org/extend/architecture
- https://www.tensorflow.org/extend/language_bindings
- https://stackoverflow.com/questions/5440968/understand-foreign-function-interface-ffi-and-language-binding
- https://kotlinlang.org/docs/reference/basic-syntax.html