Hello World প্রোগ্রাম

কটলিন কিঃ

কটলিন একটি ওপেন সোর্স, স্ট্যাটিকালি-টাইপড, মাল্টি প্যারাডাইম প্রোগ্রামিং ল্যাঙ্গুয়েজ, যা অবজেক্ট-ওরিয়েন্টেড প্রোগ্রামিং ও ফাংশনাল প্রোগ্রামিং উভয়ই সাপোর্ট করে। কটলিন প্রোগ্রামিং ল্যাঙ্গুয়েজের সিনট্যাক্স সি, সি++, সি#, জাভা প্রোগ্রামিং ল্যাঙ্গুয়েজের মত। আর কটলিন কোডে - লাইনের শেষে সেমিকোলন দেয়া অপশনাল, না দিলেও কাজ করবে।

Hello World প্রোগ্রামঃ

```
fun main(args: Array<String>) {
   println("Hello, World!")
}
Output: Hello, World!
```

এখানে main একটি ফাংশন। কটলিন প্রগ্রাম রান করে বাইট কোড তৈরি করতে JVM এর সাহায্য নেয়। আর কটলিন প্রগ্রামের এন্ট্রি পয়েন্ট হচ্ছে main ফাংশন। অর্থাৎ কটলিন প্রগ্রাম রান করে main ফাংশন থেকে। আর কটলিনে main ফাংশন লেখার পদ্ধতি হচ্ছে fun main(args: Array<String>), main ফাংশনটি আর্গুমেন্ট হিসেবে স্ট্রিং অ্যারে নিচ্ছে।

main ফাংশনের ভিতরে println("Hello, World!"), এখানে println একটি কটলিন সিস্টেম ফাংশন বা কটলিন লাইব্রেরী ফাংশন। কোনো কিছু প্রিন্ট করার জন্য print ও println ফাংশন ব্যবহার করা হয়। print ফাংশনটি সব কিছু একই লাইনে প্রিন্ট করে, অপরদিকে println ফাংশনটি কোনো কিছু প্রিন্ট করে কার্সর পয়েন্টার নতুন লাইনে চলে যায়। অর্থাৎ println ফাংশনটি, প্রিন্ট শেষে লাইন ব্রেক দেয়। print ফাংশনের মতো println ও একটি কটলিন সিস্টেম ফাংশন বা কটলিন লাইব্রেরী ফাংশন।

এখানে println ফাংশনটি "Hello, World!" কথাটি প্রিন্ট করবে। অর্থাৎ প্রগ্রামটির আউটপুটঃ "Hello, World!"

তাহলে ফাংশন কি? ফাংশন নিয়ে বিস্তারিত ধারণা পরে দেয়া হবে। এখানে শুধুমাত্র বোঝার সুবিধার জন্য ফাংশন নিয়ে সামান্য আলোচনা করা হলো।

Function is a set of instructions to perform specific tasks.
তাহলে বলা যায় কোন স্পেসিফিক বা নিদৃষ্ট কাজ করার জন্য সেট অফ ইনিস্ট্রাকশনকে বোঝায়।

```
fun name(args Arguments) {
    // Code
}
```

ফাংশনকে নেম ব্লক কোডও বলা হয়। ফাংশন আর্গুমেন্ট হিসেবে ইনপুট নিয়ে থাকে এবং সেট অফ ইনিস্ট্রাকশন বা সেট অফ কোডের মাধ্যমে আউটপুট জেনারেট করে।

Hello World প্রোগ্রামটি জাভাতেঃ

এখন দেখি Hello World! প্রগ্রামটি জাভাতে কিভাবে লেখা হতো।

```
public class Tutorial {
   public static void main(String[] args) {
       System.out.println("Hello, World!");
   }
}
Output: Hello, World!
```

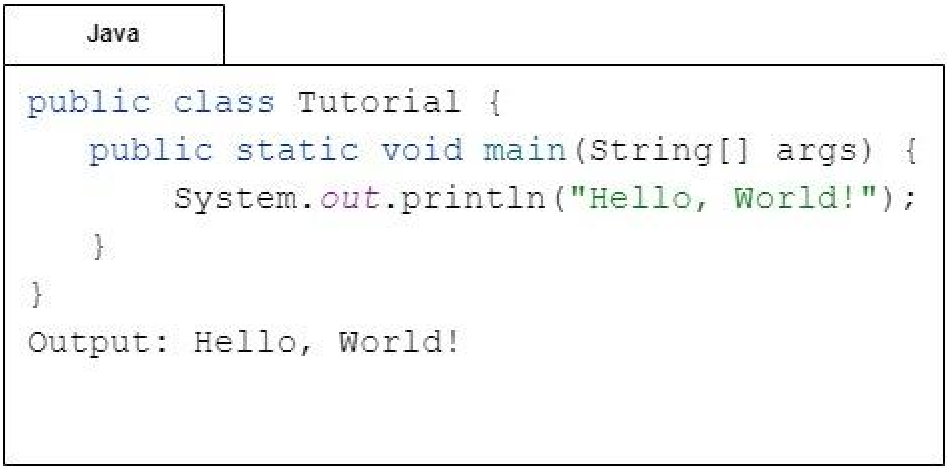
পাবলিক ক্লাস, ক্লাস নেম। ক্লাস নেম এর ভিতরে পাবলিক স্ট্যাটিক main ফাংশন বা main মেথড। জাভাতে main ফাংশন লেখার পদ্ধতি হচ্ছে public static void main(String[] args), main ফাংশনটি আর্গুমেন্ট হিসেবে স্ট্রিং অ্যরে নিচ্ছে। জাভার main ফাংশনটি একটি স্ট্যাটিক মেথড এবং রিটার্ন টাইপ হচ্ছে ভয়েড অর্থাৎ main ফাংশন কোন কিছু রিটার্ন করে না। main ফাংশনের ভিতরে System.out.println("Hello, World!"); এখানে println একটি জাভা সিস্টেম ফাংশন বা জাভা লাইব্রেরী ফাংশন। এবং লাইনের শেষে সেমিকোলন দিয়ে লাইন শেষ করা।

এখানে System.out.println("Hello, World!"); ফাংশনটি "Hello, World!" কথাটি প্রিন্ট করবে। অর্থাৎ প্রগ্রামটির আউটপুটঃ "Hello, World!"

কটলিন ও জাভা কোডের তুলনাঃ

যেহেতু কটলিন কম্পাইল হয়ে রান করতে জাভা ভার্চুয়াল মেশিন বা JVM এর সাহায্য নেয়। অপরপক্ষে আমরা সবাই জানি জাভা রান করে জাভা ভার্চুয়াল মেশিন বা JVM এর সাহায্যে। তাহলে এখন কটলিন ও জাভার কোডের কম্পাইল ও রান করা, তুলনা করে দেখা যাক।

যেকোন জাভা কোডকে কম্পাইল করে রান করতে হলে, ক্লাস বা ক্লাস ফাইলের প্রয়োজন হয়। অর্থাৎ জাভা ভার্চুয়াল মেশিন বা JVM শুধুমাত্র ক্লাস ফাইল নিয়ে কাজ করে।



```
public class Tutorial {
   public static void main(String[] args) {
       System.out.println("Hello, World!");
   }
}
Output: Hello, World!
```

```
fun main(args: Array<String>) {
   println("Hello, World!")
}
Output: Hello, World!
```

অপরপক্ষে কটলিনও জাভা ভার্চুয়াল মেশিন বা JVM এ রান করে। কিন্তু আমরা জানি কটলিনের main ফাংশন কোন ক্লাসের ভিতরে থাকে না। বা কটলিনের main মেথড রান করার জন্য কোন ক্লাসের প্রয়োজন হয় না। তাহলে এখন প্রশ্ন আসে কটলিন কোড রান করার জন্য কি রান টাইমে ইন্টার্নালি ক্লাস তৈরি কর?
এর উত্তর হ্যা।

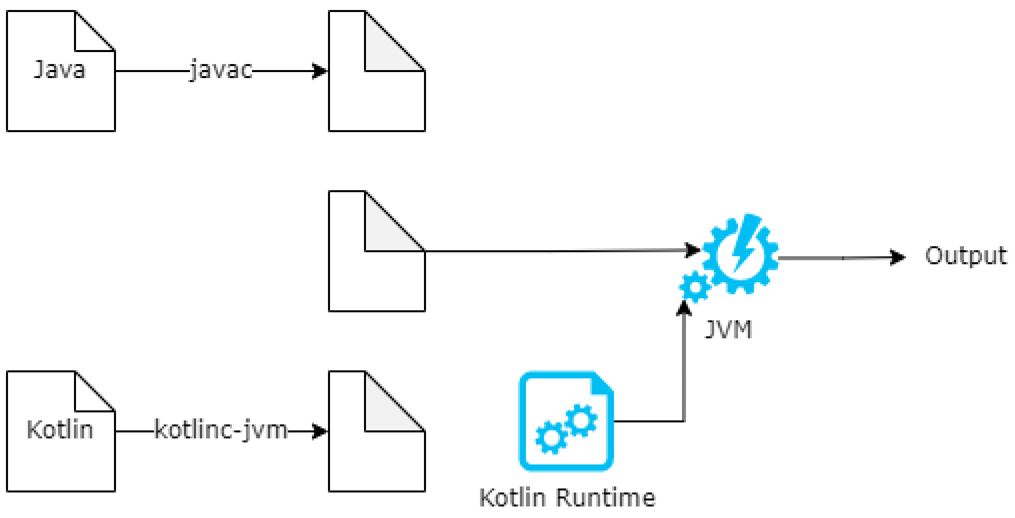
কটলিন কোড কম্পাইল করে রান করতে, রানটাইমে কটলিন কম্পাইলার ইন্টার্নালি ক্লাস ফাইল তৈরি করে মেমোরিতে লোড করে। সতরাং কটলিন রানটাইমে জাভার মতোই কাজ করে বা জাভার মতোই আচরণ করে।

কটলিন যেহেতু জাভা ভার্চ্যুয়াল মেশিন (JVM) এ রান করে সুতরাং একটা কটলিন প্রজেক্টে কটলিনের সাথে জাভা ফাইলও থাকতে পারে। অর্থাৎ একটা কটলিন প্রজেক্টে কটলিন কোড বা ক্লাস ফাইলের সাথে, জাভা ক্লাস ফাইল ও থাকতে পারে।

কটলিন ও জাভা কোড রান ও কম্পাইলেশনঃ

এখন আমরা দেখব জাভা কোড, ও কটলিন কোড কিভাবে জাভা ভার্চ্যুয়াল মেশিন বা JVM এ কম্পাইল হয়ে রান করে।

Java + Kotlin Compilation



Byte Code (.class files)

জাভা সোর্স কোড কোড JAVAC কম্পাইলে, কম্পাইল হয়ে বাইট কোড জেনারেট করে। আর কটলিন সোর্স কোড KOTLINC-JVM কম্পাইলে, কম্পাইল হয়ে বাইট কোড জেনারেট করে। এই দু রকমের বাইট কোড জাভা ভার্চ্যুয়াল মেশিন বা JVM এবং তার সাথে KotlinRuntime বা KotlinJavaRuntime মিলে আউটপুট তৈরি করে।

Agenda:

- Hello world example
- Compilation
- Bytecode

Reference:

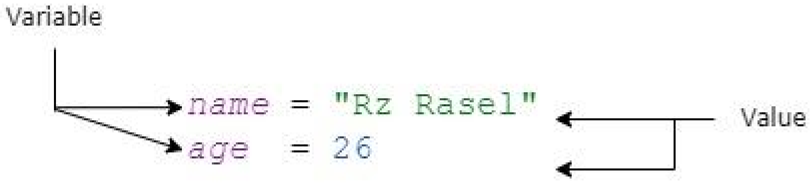
- [#2.1 Kotlin Hello World: How it works? Part-2]
- [Kotlin - Hello World Program | Compilation & ByteCode | CheezyCode #3]

End Of - Hello World!

ভেরিয়েবল কিঃ

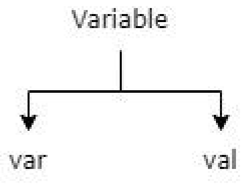
Variable in Kotlin is a data container, that saves the data values during Kotlin program execution.
ভেরিয়েবল হল একটা কন্টেইনার বা ধারক, যেখানে ডাটা বা ভ্যালু স্টোর রাখে।

যদি name = "Rz Rasel" লেখা হয় তাহলে এখানে name হলো ভেরিয়েবল এবং "Rz Rasel" হলো ভ্যালু। সুতরাং name ভেরিয়েবলের ভিতরে "Rz Rasel" ভ্যালু স্টোর করছে। একই ভাবে age = 26 লেখা হয় তাহলে এখানে age হলো ভেরিয়েবল এবং 26 হলো ভ্যালু। সুতরাং age ভেরিয়েবলে 26 ভ্যালু স্টোর করছে।



ভেরিয়েবল লেখার নিয়মঃ

কটলিনে ভেরিয়েবল লেখার জন্য var ও val দুইটি কীওয়ার্ড রাখা হয়েছে। var কীওয়ার্ডটি variable থেকে এসেছে আর val কীওয়ার্ডটি value থেকে এসেছে।



var কীওয়ার্ডঃ

```
fun main(args : Array<String>) {
   var name = "Rz Rasel"
   println(name)
}
Output: Rz Rasel
```

var name = "Rz Rasel" এখানে name ভেরিয়েবলে "Rz Rasel" ভ্যালু রাখা হয়েছে এবং তা পরবর্তী লাইনে println(name) এর মাধ্যমে প্রিন্ট করছে। এখানে name ভেরিয়েবলে রাখা ভ্যালু প্রিন্ট করছে।

```
fun main(args : Array<String>) {
   var age = 26
```

```
    println(age)
}
Output: 26
```

ঠিক একই ভাবে var age = 26 এখানে age ভেরিয়েবলে 26 ভ্যালু রাখা হয়েছে এবং তা পরবর্তী লাইনে println(age) এর মাধ্যমে, age ভেরিয়েবলে রাখা ভ্যালু প্রিন্ট করছে।

val কীওয়ার্ডঃ

```
fun main(args : Array<String>) {
    val name = "Rz Rasel"
    println(name)
}
Output: Rz Rasel
```

```
fun main(args : Array<String>) {
    val age = 26
    println(age)
}
Output: 26
```

var ও **val** এর মধ্যে পার্থক্যঃ

এখানে var এসেছে ভেরিয়েবল থেকে এবং val এসেছে ভ্যালু থেকে।

```
fun main(args : Array<String>) {
    var age = 26
    age = 27
    println(age)
}
Output: 27
```

var কীওয়ার্ড ব্যবহার করে age ভেরিয়েবল ডিক্লিয়ার করা হয়েছে এবং ভ্যালু সেট করা হয়েছে 26। পরবর্তীতে age ভেরিয়েবলের মান পূনরায় সেট করা হয়েছে 27। এবং আউটপুট হিসেবে সর্বশেষ ভ্যালু 27 প্রিন্ট হবে।

```
fun main(args : Array<String>) {
    val age = 26
    age = 27
    println(age)
}
```

```
Error: val cannot be reassigned
```

val কীওয়ার্ড ব্যবহার করে age ভেরিয়েবল ডিক্লিয়ার করা হয়েছে এবং ভ্যালু সেট করা হয়েছে 26। পরবর্তীতে age ভেরিয়েবলের মান পুনরায় 27 সেট করলে "val cannot be reassigned" এরর দেখাবে। কারণ val ভেরিয়েবলের মান শুধুমাত্র একবারই সেট করা যায়। পবর্তীতে নতুন করে মান সেট করা যায় না।

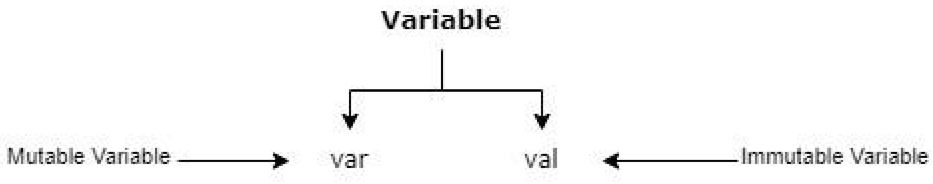
var কীওয়ার্ড ব্যবহার করে কোনো ভেরিয়েবল ডিক্লিয়ার করলে তা পরবর্তীতে যতবার ইচ্ছা ততবার ওই ভেরিয়েবলে ভ্যালু অ্যাসাইন করা যায় কিন্তু val কীওয়ার্ড ব্যবহার করে কোনো ভেরিয়েবল ডিক্লিয়ার করলে তা সম্ভব না।

val কীওয়ার্ড ব্যবহার করে কোনো ভেরিয়েবল ডিক্লিয়ার করলে, শুরুতেই তার ভ্যালু অ্যাসাইন করে দিতে হয়। কটলিনে val কীওয়ার্ডটি অনেকটা জাভার final কীওয়ার্ডের মতো কাজ করে।

Mutable ও Immutable ভেরিয়েবলঃ

var কীওয়ার্ড ব্যবহার করে ডিক্লিয়ার করা ভেরিয়েবল গুলো mutable ভেরিয়েবল। Mutable ভেরিয়েবলের মান যতবার ইচ্ছা ততবার সেট করা যায়।
var is like a general variable and can be assigned multiple times and is known as the mutable variable in kotlin.

val কীওয়ার্ড ব্যবহার করে ডিক্লিয়ার করা ভেরিয়েবল গুলো immutable ভেরিয়েবল। Immutable ভেরিয়েবলের মান একবার সেট করে দিলে আর পরিবর্তন করা যায় না বা বারবার পরিবর্তন করা যায় না।
val is a constant variable and can not be assigned multiple times and can be initialized only single time and is known as the immutable variable in kotlin.



```
fun main(args : Array<String>) {
   var number = 10 // Muttable integer
   number = 20
   println(number)
   val name = "Rz Rasel" //Immutabl/constant string
   println(name)
}
Output: 20
Output: Rz Rasel
```

প্রথমেই আমরা বলেছি, var কীওয়ার্ডটি variable থেকে এসেছে আর val কীওয়ার্ডটি value থেকে এসেছে। অর্থাৎ var বা variable এর মান চেঞ্জ করা যায় কিন্তু val বা vaule এর মান চেঞ্জ করা যায় না।

ভেরিয়েবল ডাটা টাইপঃ

```
fun main(args : Array<String>) {
   var score = 10 // Type checking
   score = "hi! Rz Rasel"
   val name = "Rz Rasel" // Type inference
   name = "Hi"
   println(score)
}
```

```
score = "hi! Rz Rasel"
```

কম্পাইল টাইমে টাইপ চেক করে এজন্য একে Type checking বলে। একে কম্পাইল টাইম এররও বলে।

```
fun main(args : Array<String>) {
   var score = 10
   var firstName = "Rashed"
   var lastName = "Uz Zaman"
   println(score)
   println(firstName)
   println(lastName)
}
Output: 10
Output: Rashed
Output: Uz Zaman
```

```
fun main(args : Array<String>) {
   var score: Int = 10 // Explicitly define variable
   var firstName: String = "Rashed"
   var lastName: String = "Uz Zaman"
   var isTrue: Boolean = true
   println(score)
   println(firstName)
   println(lastName)
   println(isTrue)
}
Output: 10
Output: Rashed
Output: Uz Zaman
Output: true
```

কটলিনে একবার কোন ভেরিয়েবেলের টাইপ ডিক্লিয়ার করলে তা আর পরবর্তীতে পরিবর্তন করা যায় না। var score: Int = 10 এখানে score একটি Int টাইপের ভেরিয়েবলের তা explicitly ডিফাইন করা হয়েছে।

ভেরিয়েবল ডাটা টাইপঃ

Byte, Short, Int, Long, Float, Double (for numbers)
Boolean (true, false)
Characters
Arrays
Strings

টাইপ ইনফারেন্সঃ

```
fun main(args : Array<String>) {
   var score: Int = 10
   println(score)
}
Output: 10
```

explicitly ডিফাইন করাকে type inference ও বলে। অর্থাৎ একই সাথে ভেরিয়েবল ডিফাইন করে তার মান নির্ধারণ করে দেয়া। var score: Int = 10 explicitly ডিফাইন করাকে type inference ও বলে। অর্থাৎ একই সাথে ভেরিয়েবল ডিফাইন করে তার মান নির্ধারণ করে দেয়া। এখানে score ভেরিয়েবল ডিক্লিয়ার করে একই সাথে তার মানও নির্ধারণ করে দেয়া হয়েছে, এইভাবে মান নির্ধারণ করে দিলে তাকে type inference বলে।

ঠিক একই ভাবে কটলিনে আগে ভেরিয়েবল ডিক্লিয়ার করে পরে মান নির্ধাণ করে দেয়া যায়।

```
fun main(args : Array<String>) {
   var score: Int
   score = 30
   println(score)
}
Output: 30
```

var score: Int এখানে প্রথমে score ভেরিয়েবল ডিক্লিয়ার করা হয়েছে, যার ডাটা টাইপ Int বা ইন্টিজার। এবং পরের লাইনে score = 30 দিয়ে score ভেরিয়েবলের মান 30 নির্ধারণ করে দেয়া হয়েছে। ফলে পরবর্তী লাইনে println(score) করলে আউটপুট হিসেবে 30 প্রিন্ট করবে।

```
var score: Int = 30
var name: String = "Rz Rasel"

fun main(args : Array<String>) {
   println(score)
   println(name)
}
Output: 30
Output: "Rz Rasel"
```

কটলিন একই সাথে অবজেক্ট ওরিয়েন্টেড ও ফাংশনাল প্রগ্রামিং সাপোর্ট করাই, var score: Int = 30 বা var name: String = "Rz Rasel" main ফাংশন স্কোপের বাইরে রাখলেও main ফাংশন থেকে score ও name ভেরিয়েবলকে এক্সেস করা যাবে।

স্ট্রিং ইন্টারপোলেশন বা স্ট্রিং টেম্পেলেটঃ

কটলিনে স্ট্রিং ইন্টারপোলেশন বা কোনকাটানেশন অনেকভাবে করা যায়। জাভার মতো প্লাস চিহ্ন দিয়েও কটলিনে স্ট্রিং কোনক্যাটানেশন করা যায়।

```
fun main(args : Array<String>) {
   var score = 100
   var firstName = "Rashed"
   var lastName = "Uz - Zaman"
   println("Hi! " + firstName + " " + lastName) // String concatenation
}
Output: Hi! Rashed Uz - Zaman
```

এখানে main ফাংশনের ভিতরে score, firstName, ও lastName নামের তিনটি ভেরিয়েবল ডিক্লিয়ার করে মান এসাইন করা হয়েছে। পরবর্তী লাইনে println("Hi! " + firstName + " " + lastName) এ প্লাস দিয়ে স্ট্রিং কোনক্যাটানেশন করা হয়েছে।

```
fun main(args : Array<String>) {
   var score = 100
   var firstName = "Rashed"
   var lastName = "Uz - Zaman"
   println("Hi! ${firstName} ${lastName}, your score is: ${score}")
}
Output: Hi! Rashed Uz - Zaman, your score is: 100
```

println("Hi! ${firstName} ${lastName}, your score is: ${score}") এখানে ডলার সাইন সেকেন্ড ব্রাকেট বা কার্লি ব্রাকেটের ভিতরে firstName, lastName ও score ভেরিয়েবল বসিয়ে স্ট্রিং টেমপ্লেটিংএর মাধ্যমে স্ট্রিং কোনকাটানেশন করা হয়েছে।

```
fun main(args : Array<String>) {
   var score = 100
   var firstName = "Rashed"
   var lastName = "Uz - Zaman"
   println("Hi! $firstName $lastName, your score is: $score")
}
Output: Hi! Rashed Uz - Zaman, your score is: 100
```

println("Hi! $firstName $lastName, your score is: $score") সেকেন্ড ব্রাকেট বা কার্লি ব্রাকেট তুলে দিয়েও সর্ট পদ্ধতিতে স্ট্রিং টেমপ্লেটিং করা যায়।

```
fun main(args : Array<String>) {
   var score = 100
   var firstName = "Rashed"
   var lastName = "Uz - Zaman"
   var name = "$firstName $lastName"
   println("Hi! $name, your score is: $score")
}
Output: Hi! Rashed Uz - Zaman, your score is: 100
```

এখানে var name = "$firstName $lastName" এখানে ডলার সাইন ব্যবহার করে স্ট্রিং টেম্পেলটিংএর মাধ্যমে name ভেরিয়েবলের মান সেট করা হয়েছে। এবং পরের লাইনে println ফাংশনকে $name ও $score স্ট্রিং টেমপ্লেটিংএর মাধ্যমে স্ট্রিং ইন্টারপোলেশন বা কোনকাটানেশন করা হয়েছে। আউটপুট হিসেবে এখানে Hi! Rashed Uz - Zaman, your score is: 100 প্রিন্ট করছে।

Agenda

Reference:

- [#3.2 Kotlin Variables and Data Types. Kotlin Basic Syntaxes]
- [Variables & Data Types in Kotlin Tutorial | var & val | CheezyCode #4]

End Of - Variable
end

Referance:

- Kotlin for Beginners - Part 4 - VARIABLES
- Variables and Basic Types in Kotlin - Android Studio Tutorial
- Variables in Kotlin - Kotlin Tutorial for Beginners

End - Variable

List of content

- Basic Syntax Comments
- Function
- Class
- String interpolation
- Loop
- Variables and Constants
- Data Types: var vs val

#4.2 Kotlin Data Types: VAR vs VAL: Variables and Constants Part-1
https://youtu.be/qAJTqI_aKJU?list=PLIxmoA0rQ-LwgK1JsnMsakYNACYGa1cjR

List of content

- Variables
- Basic data types
- Arithmetic operators

Kotlin for Beginners - Part 4 - VARIABLES
https://youtu.be/kYO7YWVB3jY?list=PLrnPJCHvNZuAIbejjZA1kGfLeA8ZpICB2

List of content final
- Hello World! How does it work?
- Basic syntax
- Variables
- Basic data types
- Arithmetic operators

KotlinConf 2017 - Introduction to Coroutines by Roman Elizarov
KotlinConf 2017 - Deep Dive into Coroutines on JVM by Roman Elizarov

Rz Rasel