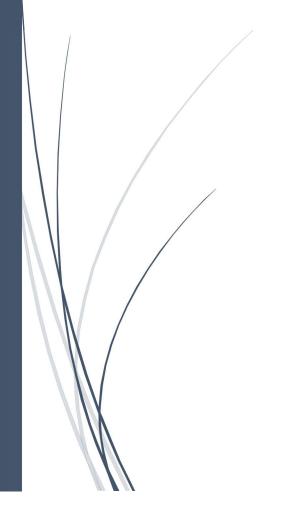
Kode

Dev Example Guide



Arpan Mahanty EDUMATE

1 PRINT STATEMENT

```
The print statement works just like python. Syntax: print(*str, sep = ' ', end = '\n');

Example — print('Hello World'); print(a); // Let a be 5.025 print('My age is', age); // Let age be 18 print(1,2,3,4,5, sep="-", end="Bye")

Output — Hello World 5.025
My age is 18 1-2-3-4-5Bye
```

2 DATA TYPES

Number(), String(), Bool(), List() Equivalent to int() / float(), str(), etc. in python

2.1 VARIABLE DECLARATION AND ASSIGNMENT

```
Example –
var age = 20;
var a,b = "Arpan",c;
a = 20;
c = True
```

Note – By default variables initialize as None.

2.2 IDENTIFIERS

For identifier naming you can use A-Z, a-z, 0-9 and _

2.3 FORMATS

2.3.1 String

String formatted data can be written as both "Example Text" and 'Example Text' format. Use String() to convert any data-type to string format.

2.3.2 Number

Number can be entered either in integer / decimal / scientific format (or notation) Example – 5, 10.02, -10.25, 1.65E7, 1.67e-10, etc.

2.3.3 Bool

Bool type can have only values as 'True' or 'False' which also acts equivalent to numeric 1 or 0. Example – True, False
True + True is equal to 2
True – False is equal to 1

2.3.4 List

List are mutable in nature and can have dynamic length. var a = [1, 2, "Hi", True, None];

2.4 SOME SPECIAL NOTATIONS

None equivalent to null in Java or None in python Infinity equivalent to ∞
NaN equivalent to undefined

3 OPERATORS AND EXPRESSIONS

```
3.1 ARITHMATIC
      // Addition
a+b
       // Substraction
a-b
a*b
      // Multiplication
       // Division
a/b
a%b
       // Mod or Modulo Division
a\b
      // Integral Division
       // Unary Plus
+a
       // Unary Minus
-a
var expr = (a+b)*(c+d/e)\f;
3.2 COMPARISON
a==b, a!=b, a<b, a>b, a<=b, a>=b
You can compare about almost everything
3.3 LOGICAL
!True // not True
!False
True or False
True and False
```

4 IMPORT STATEMENT

```
Same as Python but you can not use '*'
import module.name;
import module.name as allies;
from module.name import var1, var2;
```

5 CONTROL FLOW

5.1 IF-ELSE STATEMENT

```
}
5.2 FOR LOOP
for (var a = 1; a < 10; a = a + 1) {
 print(a);
5.3 WHILE LOOP
var a = 1;
while (a < 10) {
 print(a);
 a = a + 1;
}
    FUNCTIONS
Functions can also be treated as values ( or literals )
makeBreakfast(bacon, non_veg = eggs, toast);
makeBreakfast();
fun printSum(a, b) {
 print(a + b);
 return a+b;
var sum = printSum;
sum(1, 2);
Another Example -
fun returnFunction() {
 var outside = "outside";
 fun inner() {
  print outside;
 }
 return inner;
var fn = returnFunction();
fn();
Functions works like in python having keywords like argument structure
fun identifier(key = default_value, ... ){
        body;
}
```

7 COMMENTS

// Single Line Comments

/* Multi Line Comments

8 CLASS

```
class Breakfast {
 __init___(meat, bread) {
  this.meat = meat;
  this.bread = bread;
 }
// ...
var baconAndToast = Breakfast("bacon", "toast");
baconAndToast.serve("Dear Reader");
// "Enjoy your bacon and toast, Dear Reader."
__init__ is initializer / constructor
__str__ is for string representation
__bool__ is for boolean representation
__num__ is for numeric representation
__list__ is for list representation
class Brunch < Breakfast {</pre>
 drink() {
  print "How about a Bloody Mary?";
 }
}
Here Brunch is class name ( or Sub-Class Name ) and Breakfast is Super-Class Name.
var benedict = Brunch("ham", "English muffin");
benedict.serve("Noble Reader");
class Brunch < Breakfast {</pre>
 __init__(meat, bread, drink) {
  super.__init__(meat, bread);
  this.drink = drink;
 }
}
```

9 TRY-EXCEPT

```
try{
        // statements
} except {
        print("Error has occurred");
}
try{
        // statements
} except(Error) {
        print("Error has occurred");
} except(SomeOtherError){
        print("Error has occurred");
}
try{
        // statements
} except(Error as e) {
        print("Error has occurred:", e);
}
```

Here 'e' contains the instance of the error class.

Can not handle non-runtime errors.

10 Note

- It still lacks lot of libraries
- There are some functions available like only print, exit, etc.

You can also follow the syntax of jlox programming language as Kode has been derived from jlox. https://www.craftinginterpreters.com/the-lox-language.html