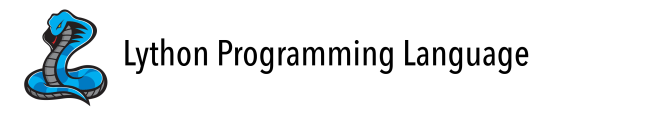
Lython Programming Language - Non-indented Python

Gabriel Margarido - February 12th 2023

Inspired on and influenced by: Solidity, VisuAlg, Ruby, Lua, Java, Javascript and C



* Build Lython Compiler from sources with Makefile or make.bat:

cd lython-sources && make

* Compile source-code using LythonC

lythonc <file>.ly

* Basic algorithm structure

algorithm

function main()

# "Your code goes here"

end

endalgorithm

* Basic "Hello world"

algorithm

function main()

print("Hello world")

end

endalgorithm

* X-times loop

algorithm

function main()

5 times do

print("Doing...")

end

end

endalgorithm

* Local-global variable (i) represents the current iterator value (integer).

5 times do

String index = str(i)

print("Number: "+index)

end

* Local-global variable (i) represents the current iterator value (integer).

int a = 3

if (a is 3) do

...

elseif (a >= 3) do

...

elseif (a isnot 3) do

...

else

...

end

is -> (==)

isnot -> (!=)

* Variable declaration

int a = nil

float b = nil

String c = nil

bool d = nil

mathematical e = nil

* Variable initialization

int a = 4

float b = 34.6

String c = "Hello world"

bool d = True

mathematical e = "(13+(45/4))/b+a"

* Variable reassignment

a := 56

b := 45.1

c := "Bye World"

d := False

e := "((a+b)/b)+24.5"

* Array initialization

int[] a = (0, 5, 10, 15, 20, 25)

float[] b = (34.6, 78.2, 0.50, 0.01)

String[] c = ("Hello world", "Bye World", "Go on!")

* Array reassignment

a := (1, 2, 3, 4, 5)

b := (45.1, 78.9015, 456.90)

c := ("Bye World","This is Sparta","Etcetera")

* Declaring functions

function myfunc(a, b, c, d, e)

mathematical f = "((a+b)/c)\*e+d"

return f

end

* Calling functions with return

float u = myfunc(a, b, c, d, e)

print(u)

* Calling Python 3 functions

You can just call Python's functions as they are.

print("Hello world")

String x = input()

String x = input()

int k = int(x)

String x = input()

float k = float(x)

* Loops

for i in 0..100 do

print("Hey from 0 to 100, this is: "+i)

end

while (True) do

print("Repeating for ever")

end

5 times do

print("Repeating 5 times")

end

* Try-Catch-Finally Error handling

try

print("Hello")

catch

print("Hello world")

finally

print("Bye")

end

* Importing external code

Import all files inside a package, same as: from tkinter import \*

@include\_module tkinter

Import a single source-code file, same as: import os

@include os

* Still in development - Creating Classes

class Motorbike do

function constructor(this) do

# "Your code here"

end

end

* Still in development - Creating Classes with single-inheritance

class Car < Motorbike do

function constructor(this) do

# "Your code here"

end

end

* Still in development - Creating Classes with multiple-inheritance

class Bus < (Motorbike, Car) do

function constructor(this) do

# "Your code here"

end

end

* Still in development - Auto-reference to parent class

class Car < Motorbike do

function constructor(this, model, age, register) do

String model = this.model

int age = this.age

String register = this.register

end

end

* Still in development - Adding methods to class

class Car < Motorbike do

function constructor(this, model, age, register) do

String model = this.model

int age = this.age

String register = this.register

end

function brake() do

print("Stopping....")

end

end

* Still in development - Calling methods from class

@new hyundai : Car("HB20", 2018, "KPX-3088")

hyundai.brake()

* Still in development - Passing class

class Car < Motorbike do

function constructor(this) do

pass

end

end

* Extra Values

True

False

nil (Same as None)

You can also call all Python functions from their implementation as they are.

JPython, IronPython, GraalVM or CPython.

Visual Studio Code Extension available at: vscode/lython-syntaxhighlight

OS X or Linux: Drag "lython-syntaxhighlight"and drop into "~/.vscode/extensions".

Windows: Drag "lython-syntaxhighlight"and drop into "%USERPROFILE%\.vscode\extensions".