

ghPython101

introducción a la programación
en python para Grasshopper

Fablab Sevilla + Universidad de Sevilla + Ángel Linares.

*Aula Star.
Departamento de
Estructuras de Edificación.
ETSAS.
1000 - 1400*

*Jueves Nov 5
Martes Nov 10
Jueves Nov 12
Martes Nov 17
Jueves Nov 19*

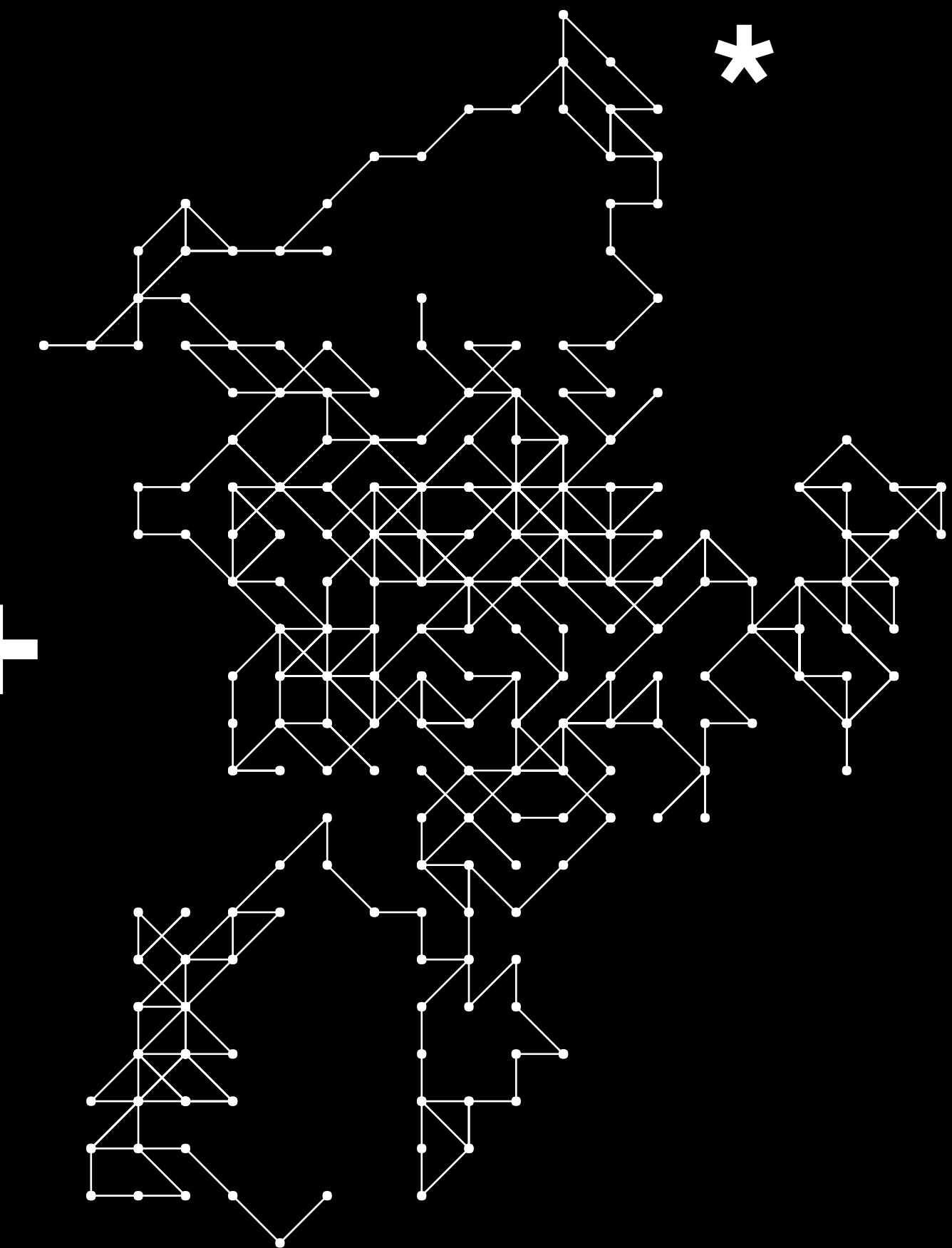


Buenos días!!

ángel linares
freak de la geometría y la programación amateur
enreda tecnológico

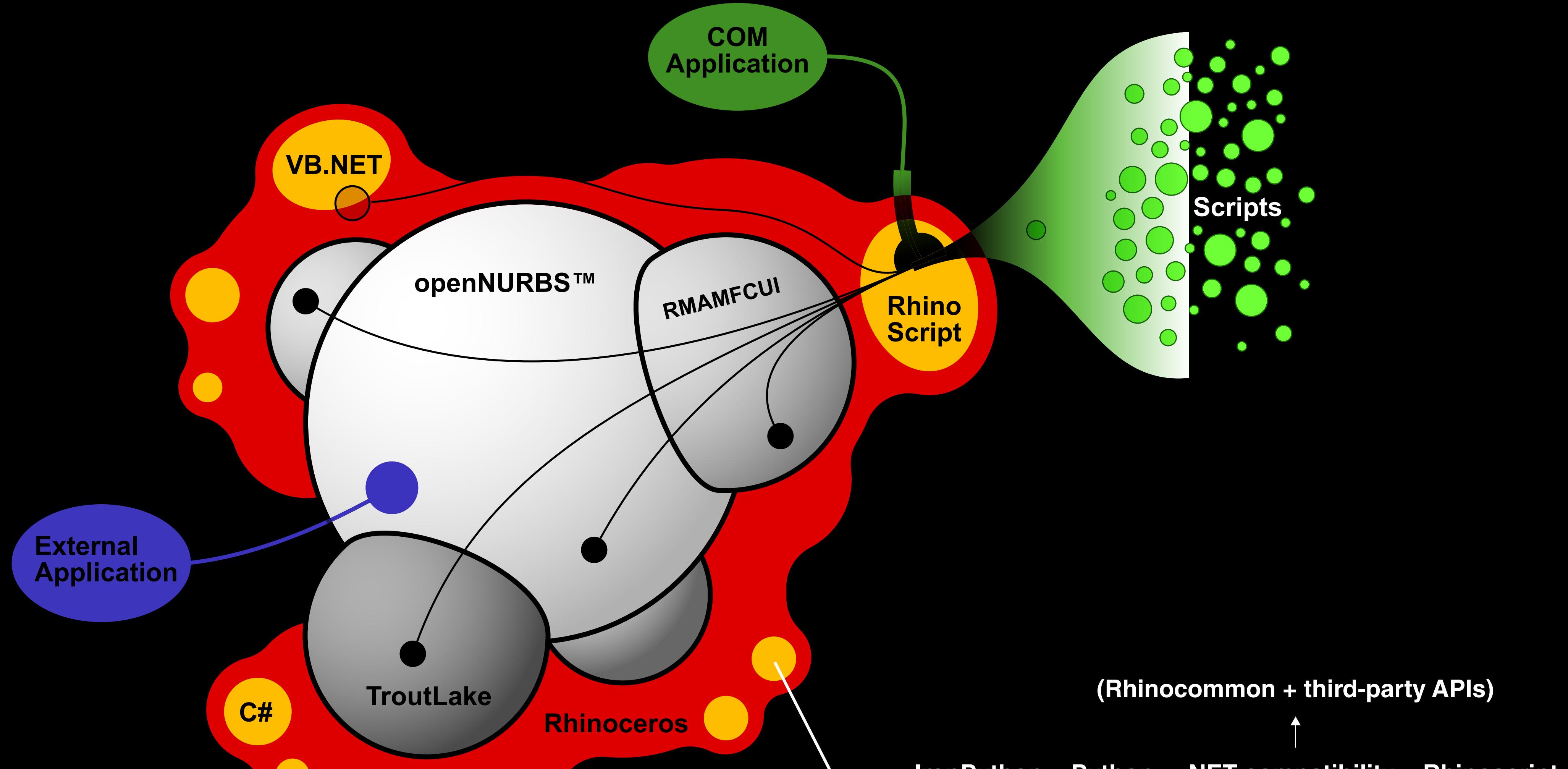
*@_aLinG_
www.angellinares.xyz // www.blurrypaths.com
contact@angellinares.xyz*

A +



*

* Cosas raras: más info en [linkedin](#) o [blurrypaths](#)



from RhinoPython101 Primer (page 16)

`import this`

The Zen of Python

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one-- and preferably only one --obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

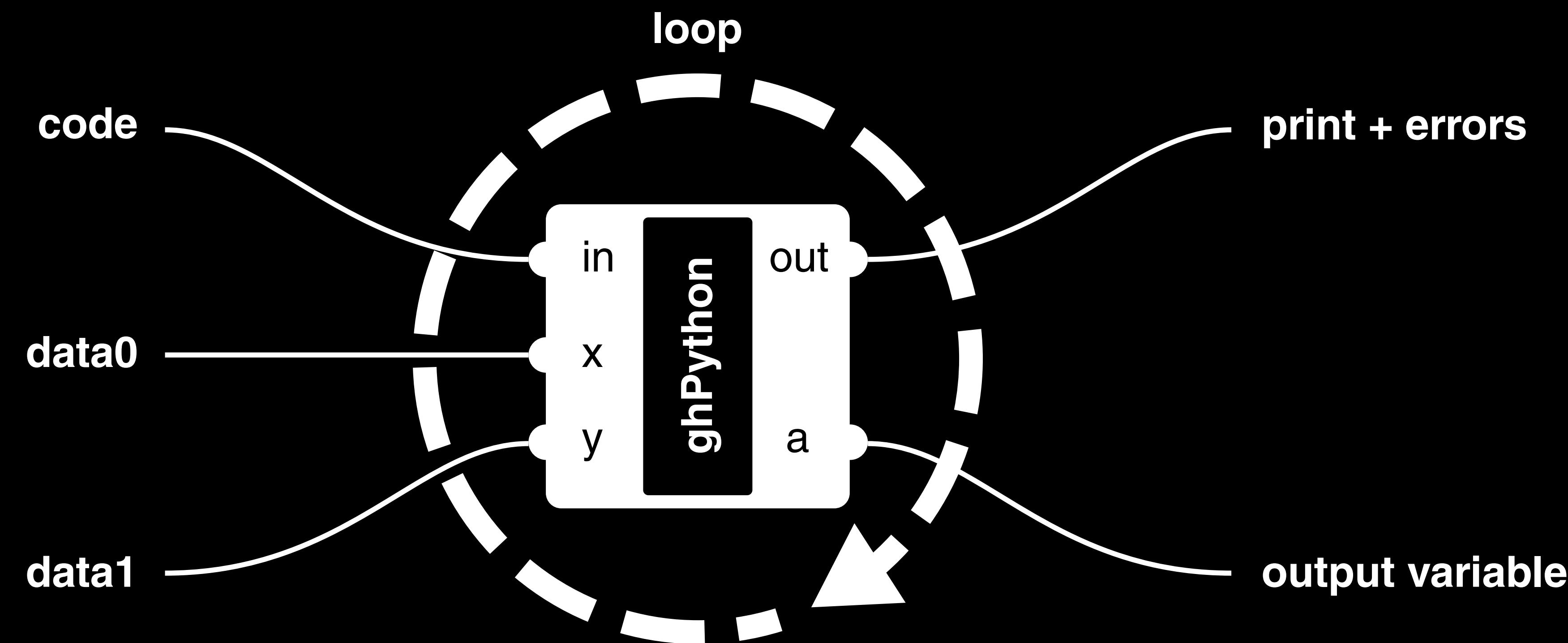
Now is better than never.

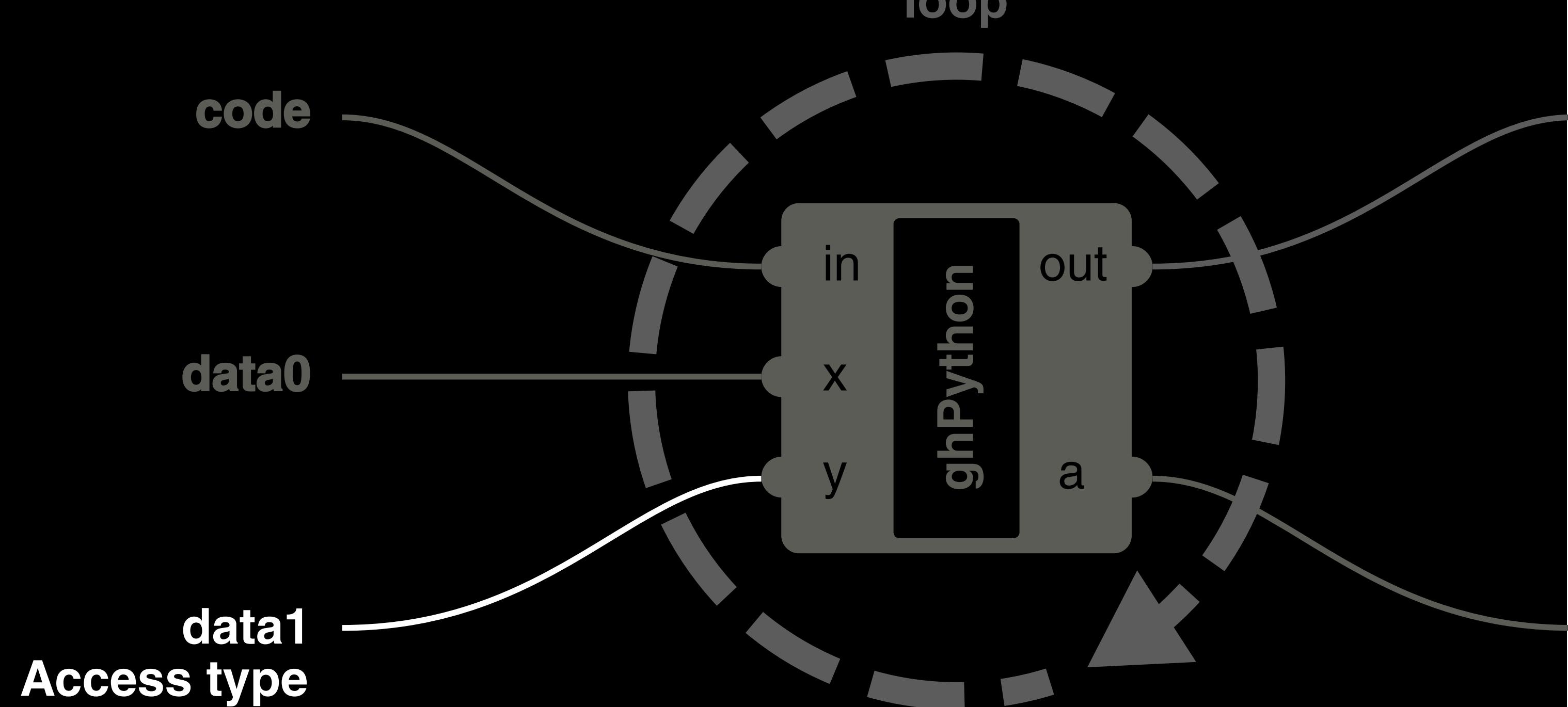
*Although never is often better than *right* now.*

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!

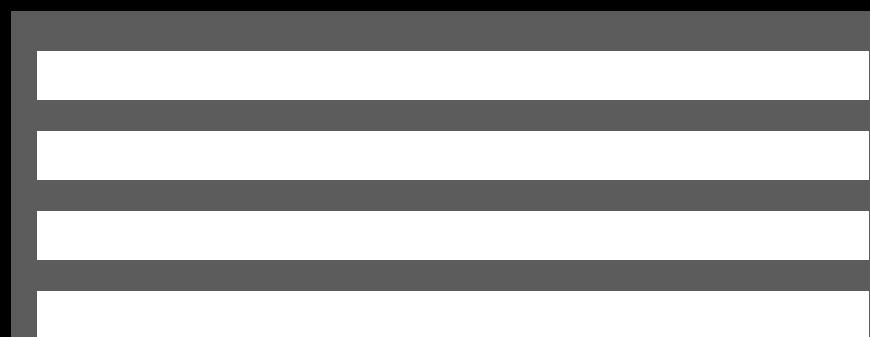




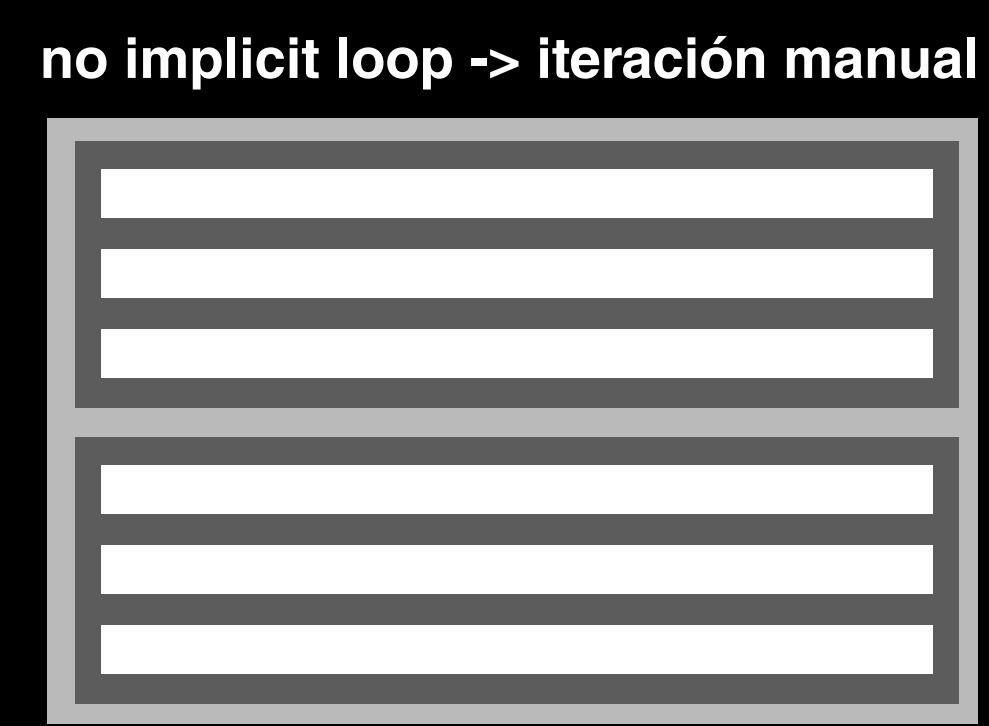
`y = item` `y = list`

→ implicit loop

implicit loop en el arbol de datos



`y = tree`



```
# Esto es un comentario, Python no lo lee  
# Ojo con las tildes y “enies” -> Mejor English  
  
strName = “angel” # ...y esto una variable
```

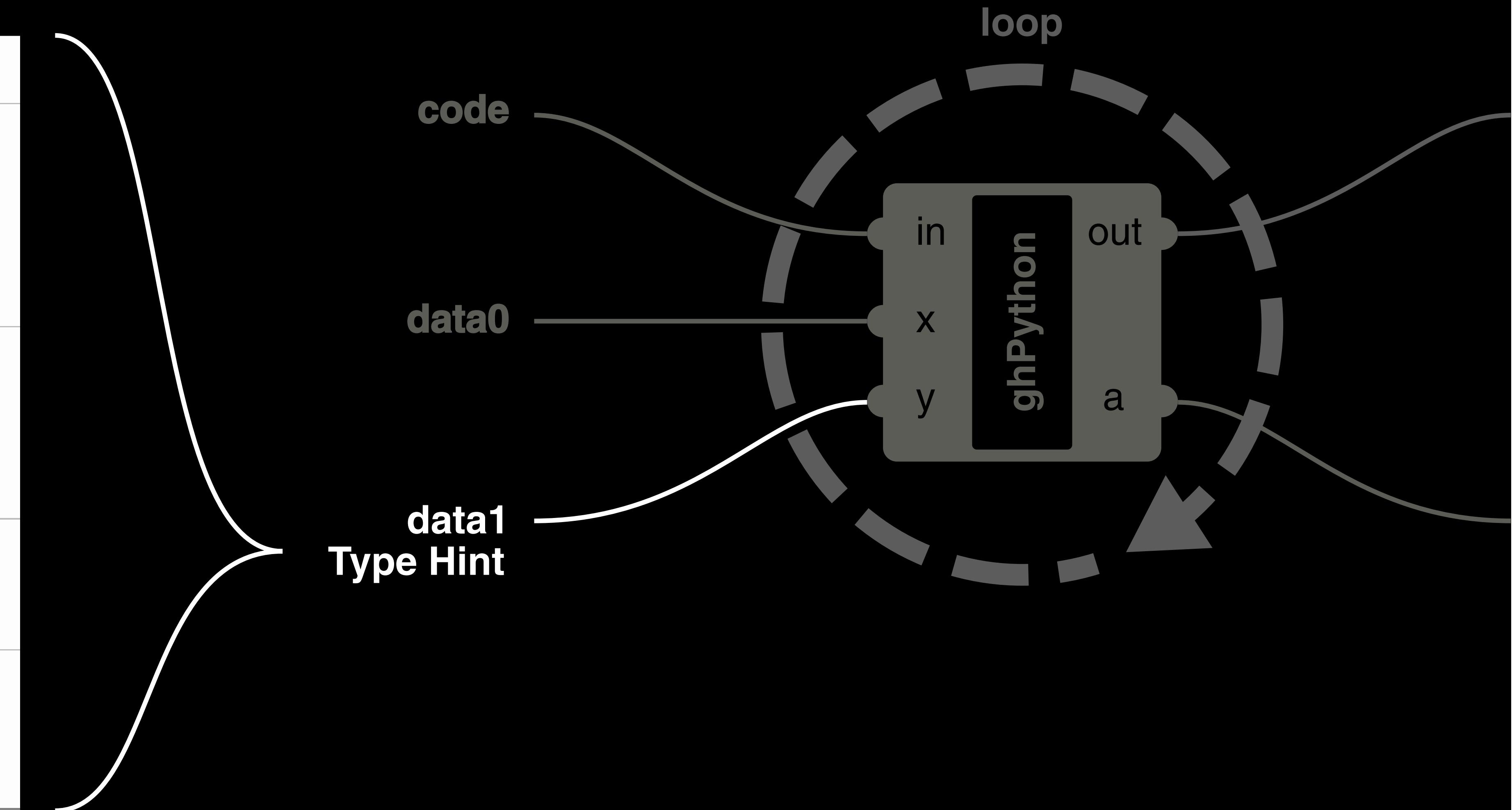
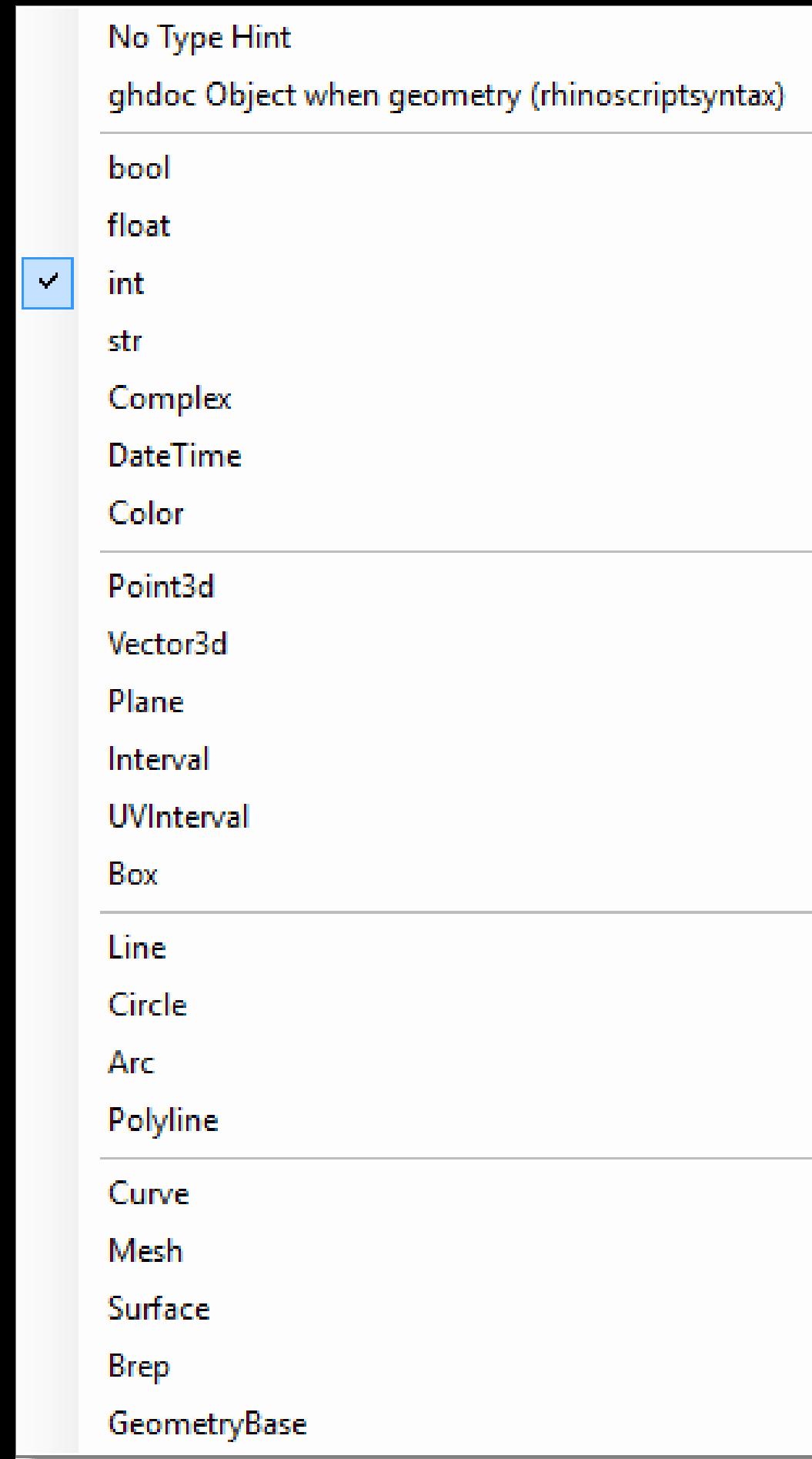
```
# Python es case sensitive

strName = "angel"
strLastName = "linares" # string || buscar "estrinque" en RAE
intAge = 32              # integer
fHeight = 1.82            # float
bMarried = False          # boolean
print strname             # debe dar un error.

# Esto es una lista

listHobbies = ["geometría", "programación", "tecnología"]

listHobbies[0] -> "geometría"
listHobbies[-1] -> "tecnología"
```



```
# Construcción de una lista \\ mirar documentación Python

listA = []                      # lista vacia
listB = [0,1,2,3,4]              # lista llena especificando valores

listMajara = ["ghPython", 101, 1000.0, True, ["a","b"]]

listA.append(5)                  # adding one new element
listB.extend(listA)              # adding listA to listB

newList = listB[1:3]              # cortamos la lista del indice 1 al 3
 newList1 = listB[:]              # copia de la lista
 newList2 = listB                # nueva instancia de listB
 newList3 = listB[1::2]            # secuencia 1 = inicio, 2 = salto
```

Estructura condicional de linea simple

```
if a == b: a+=1  
    condición     acción
```

Estructura condicional de linea múltiple

```
condición  
if a == b:  
    a+=1 | acción  
    b-=1
```

Estructura condicional multi-condición

```
if condición:  
    acción  
else:  
    acción2  
  
if condición:  
    acción  
elif condición2:  
    acción2  
else:  
    acción3
```

Condiciona

Estructura de bucle simple for.

```
for i in range(10): #mirar función range // xrange  
    contador  
    print i  
    acción
```

Estructura de bucle con iterable.

```
for pt in pts:  
    variable de paso      iterable  
    print pt.x  
    acción  
        for i, pt in enumerate(pts):  
            if i < len(pts)-1  
                pts[i] = pt+pts[i+1]/2
```

Estructura bucle while

```
while area<target:  
    condición  
    scale(curve,1.01)
```

Iterandooo!

Construcción de funciones

```
def strNombreFunc (strVarA, strVarB):  
    nombre           variables de entrada | scope  
    return strVarA + strVarB  
comando de devolución
```

Las variables de entrada son opcionales.

¿Qué es el “scope”?

Las funciones podrán devolver o no un resultado.

Objetivo de las funciones:

1.- compactar el código

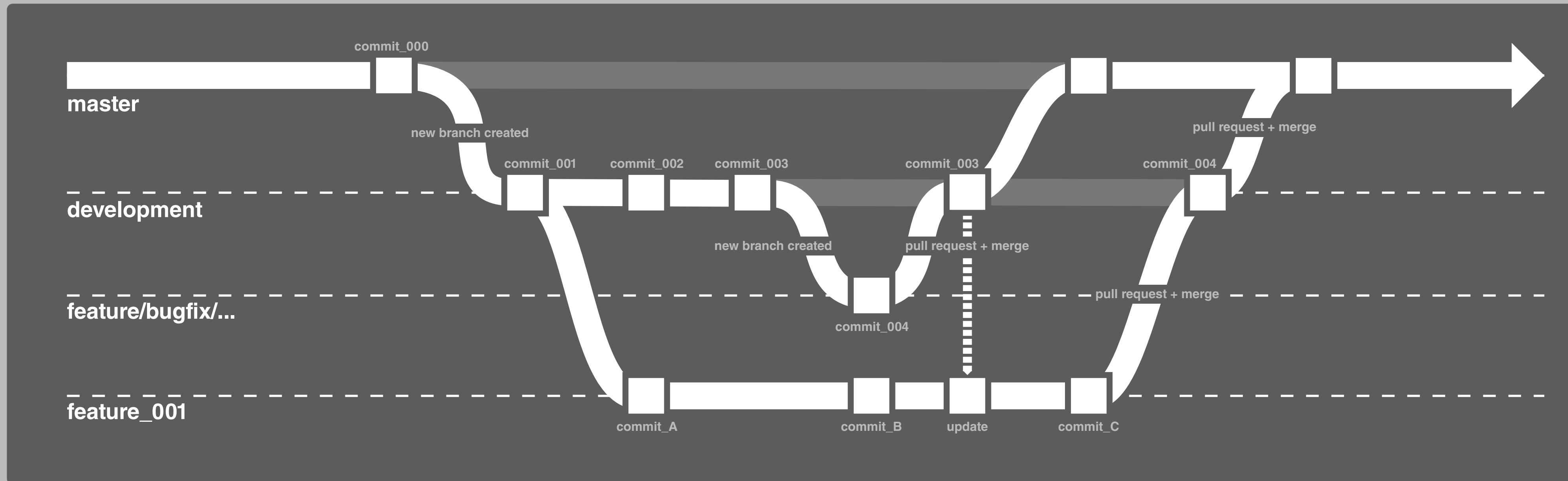
2.- generar una nueva capa de abstracción

3.- facilitar el mantenimiento del código

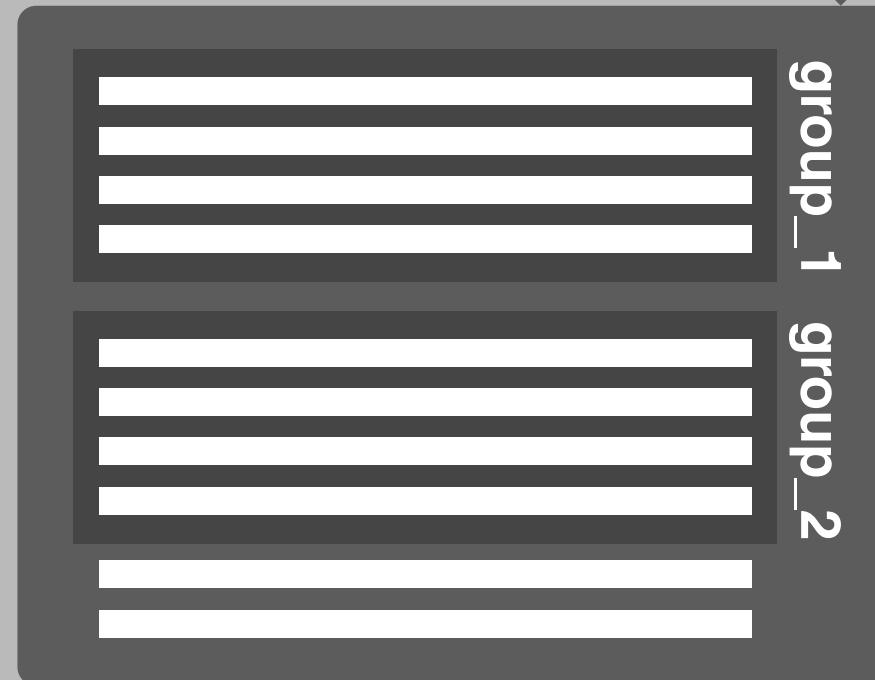
4.- permitir nuevas “trucos” algorítmicos -> recursión

repository

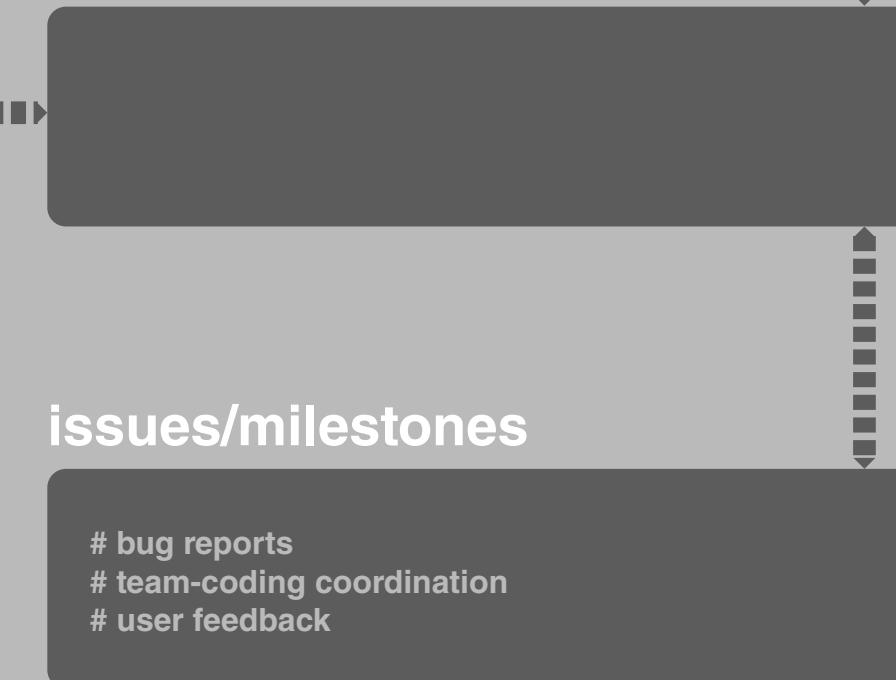
data



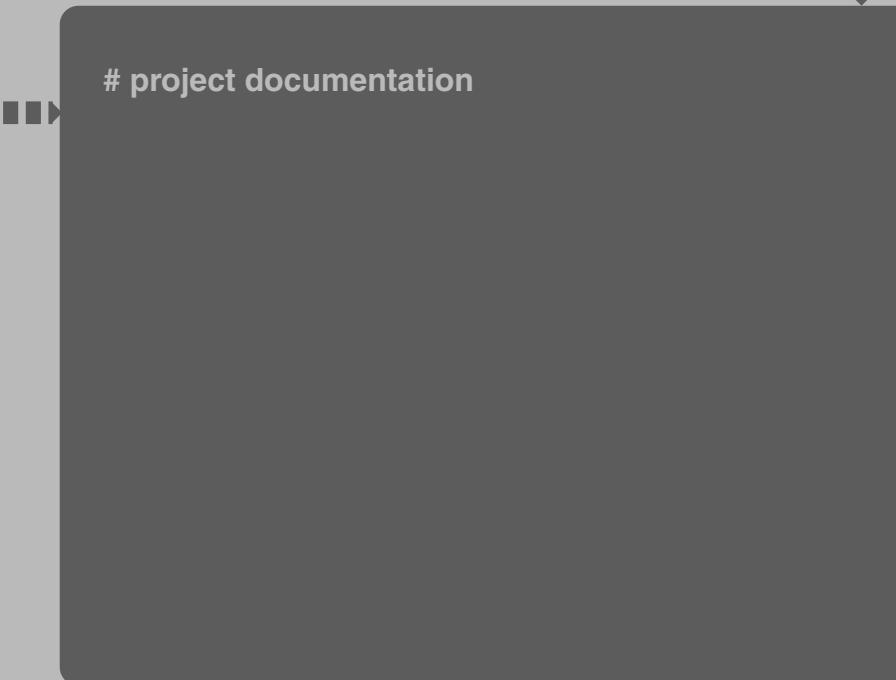
collaborators



statistics



wiki



pages

