COMPUTATION UNIT

Advanced Design Studies

The University of Tokyo

RHINO PYTHON LECTURE | DAY1

1. Intro

1.1. What's Python?

There is so many languages for programing.

C, C++, C#, Java, Scala, Go, Rust, Ruby, Python …etc

We can divide them into 2 types.

Compile Language

Before executing code, they need 'compile' that translates the code into machine language like 0110001…

The program written in Compile Language run fast. C, C++, C#, Java, Scala, Go and Rust are here.

Scripting Language

Scripting Language doesn't need 'compile' to execute.

(Actually it's compiled in the interpreter, but we can't see the process)

We can execute the code written in Script Language while writing it.

Ruby and Python are this type.

So that mean Script Language like Python make it faster to check and revise the code.

1.2. What's RhinoPython?

We can use Python in Rhinoceros and Grasshopper.

You may need Python if you want to:

- Automate a repetitive task in Rhino much faster than you could do manually.
- Perform tasks in Rhino or Grasshopper that you don't have access to in the standard set of Rhino commands or Grasshopper components.
- Generate geometry using algorithms.
- Many, many other things. It is a programming language after all.

2. Editor

2.1. IDE (Integrated Development Environment)

free

Atom : *https://atom.io/*

VS Code : *https://code.visualstudio.com/*

Anaconda : *https://www.anaconda.com/*

fee

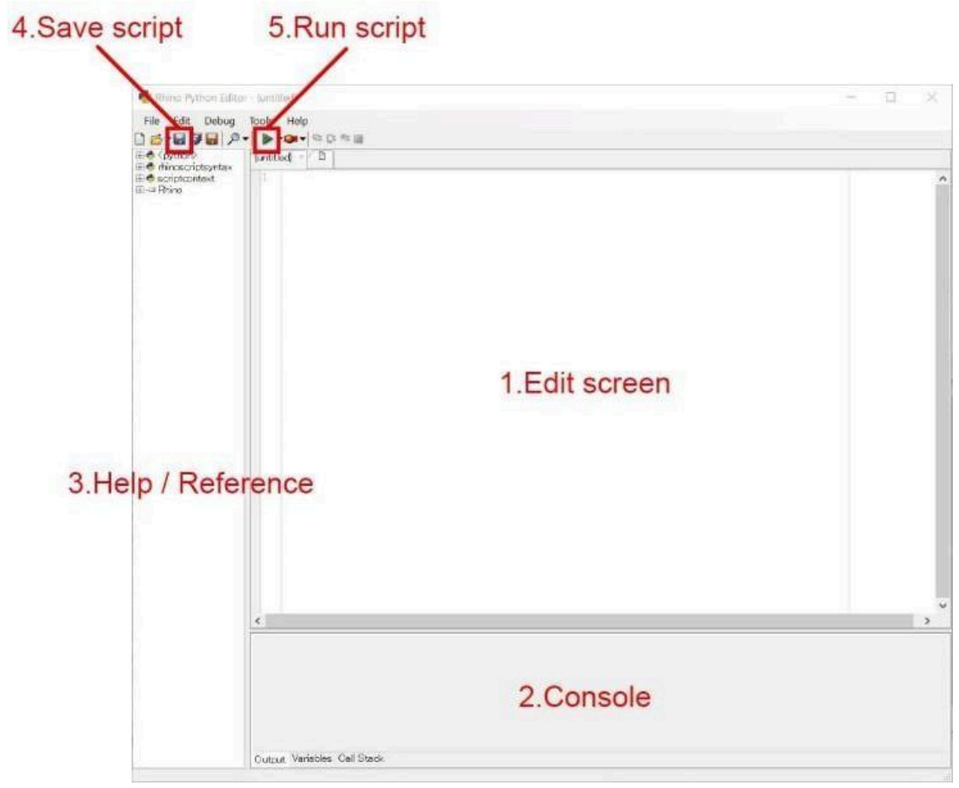
PyCharm : *https://www.jetbrains.com/pycharm/*

2.2. RhinoPython Editer

Please enter following characters at Rhino's command line —

```
EditPyrthonScript
```

2.2.1 User Interface



Let Try

A. Write following code into the edit screen

```
a = 1
b = 2 print
a + b
```

B. Write following code into the edit screen

```
import rhinoscriptsyntax as rs
plane = rs.WorldXYPlane()
rs.AddArc( plane, 5.0, 300.0 )
```

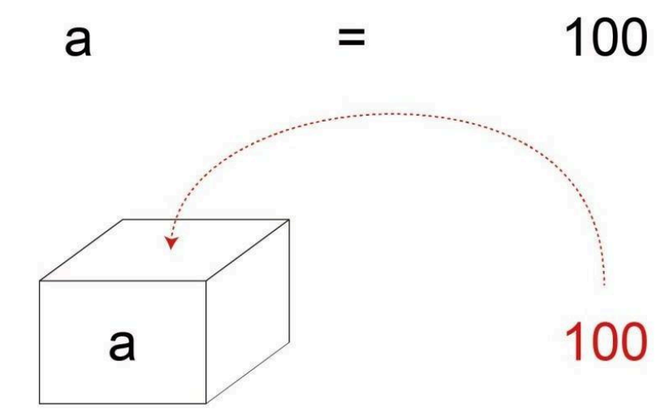
---

3. Variables

3.1. Substitution

Programs are series of procedure to deal with bunch of variables.

Variables are the main elements in computer programs and are sometimes compared to boxes to store data. Every variable have its data type.



Variables are sometimes compared to boxes to store data.

"=" is a relational operator that indicates right-hand side and left-hand side are equal in mathematics, but is a operator to substitute right-hand side for left-hand side in programming.

4. Data Type

Each value has their own data type in program.

Data Type	Example
Integer	1, 30, 1111
float	0.11111, 1.223

Data Type	Example
string	ABCD, text, A112k
boolean	true, false
list	[1, 1, 3, 30]

4.1. Check Variables and Data Type – Print statement

You can check the value of variables with ‘print’. It make us see the value on the console panel.

```
#Check variables by printing at console.

#int - Integer variables
i = 1
print i
print type(i)

#float - floating-point arithmetic
f = 0.001
print f
print type(f)

#str - Text variables
#you can use single quotation or double quotation, whichever you want. s = "text" print s #bool - True/False variables (binary values) b1 = True b2 = False print b1, b2 s =
"text"
print s
print type(s)

#list(array) - list(array) variables
l = [1,5,"text"]
print l
```

```
print type(1)
```

5. Calculation

5.1. Four arithmetic operations

```
#Description method of arithmetic operations:

#Assume variable x holds 10 and variable y holds 4,
x = 10
y = 4

#Add two numeric values
add = x + y
print add

#Subtract two values
sub = x - y
print sub

#Multiply two values
mul = x * y
print mul

#Divide two values
div = x / y
print div

#Divide two numbers and return only the remainder
rem = x % y
```

```
print rem


#Use "()"-brackets
b1 = x + y * 2
b2 = (x + y) * 2
print b1
print b2


```

5.2. Practice

5.2.1. Editing Variables and Arithmetic Operations

Find the solution of mathematical formulas by defining Variables and Arithmetic Operations.

#Example

$$55 \times 25 - 59 \times 25$$

Answer

#Exercise1

$$60 \times \left(\frac{7}{12} - \frac{11}{15} \right)$$

Answer

#Exercise2

$$27 \div \{1 - 2 \times (1 - 5)\}$$

Answer

---

6. If

Conditional Branch – To branch processes in programming, use “if… elif… else…” syntax. The space before lines next to the “if”, “elif”, “else” lines are necessary to define paragraph in program.

```
If  {conditional sentence 1} :
    {If  statement line1}
    {If  statement line2}
elif {conditional sentence 2} :
    {If  statement line1}
    {If  statement line2}
        .
        .
        .
```

Use 4 spaces per indentation level

There are several relational operators—“Comparison operator”, “Logical operator”

6.1. Comparison operator

- > – larger(greater) than
- < – smaller(less) than
- == – equal to
- != – not equal to
- >= – larger(greater) than or equal to
- <= – smaller(less) than or equal to

?

```
################################################
#Comparison operator
################################################

x = 0

if x > 5 :
  a = "x is higher than 5"
elif x < 5 :
  a = "x is lower than 5"
else :
  a = "x is 5"

```

6.2. Logical operator

- and – (a and b) is true
- or – (a or b) is true
- not – Not(a and b) is false.

```
################################################
#Logical operator - Two conditional expressions
################################################

```

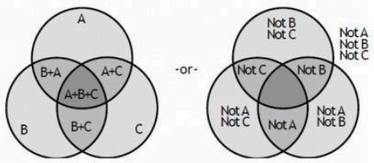
```
a = 0
b = 1

#If both the operands are true then condition becomes true. if
a == 0 and b == 1:
    print "(a and b) is true"

#If any of the two operands are true then condition becomes true. if
a == 0 or b == 1:
    print "(a or b) is true"

#Used to reverse the logical state of its operand.
if not(a == 0 and b == 1):
    print "Not(a and b) is true."

################################################
#Logical operator - Three conditional expressions
################################################

a = "a"
b = "a"
c = "a"

if a == "a" and b == "b" and c
    =="c": print "(a and b and c)
    is true"

if a == "a" or b == "b" or c
    =="c": print "(a or b or c)
    is true"
```

```
32      if not(a == "a" and b == "b" and c
33          =="c"): print "Not(a and b and c) is
34          true."
35
```

6.3 Practice

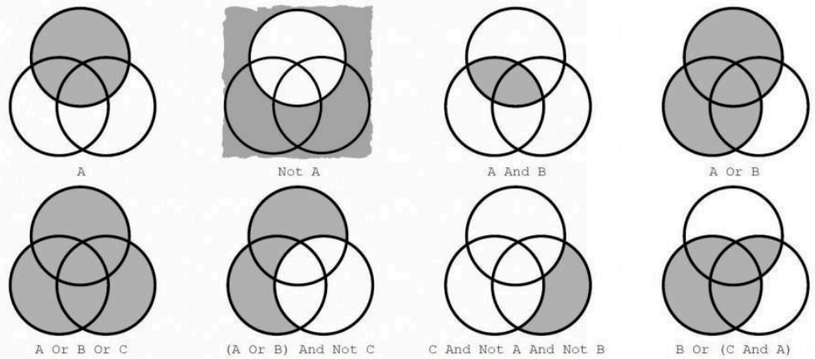
6.3.1.Exercise

A good way to exercise your own boolean logic is to use Venn-diagrams. A Venn diagram is a graphical representation of boolean sets, where every region contains a (sub)set of values that share a common property. The most famous one is the three-circle diagram:

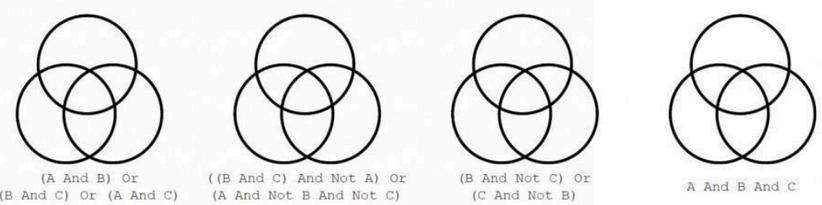


-or-

Every circular region contains all values that belong to a set; the top circle for example marks off set {A}. Every value inside that circle evaluates True for {A} and every value not in that circle evaluates False for {A}. If you're uncomfortable with "A, B and C", you can substitute them with "*Employed*", "*Single*" and "*HomeOwner*". By coloring the regions we can mimic boolean evaluation in programming code:



Try to color the four diagrams below so they match the boolean logic:



7. Iteration

7.1. For Loop Statements

For loops are traditionally used when you have a piece of code which you want to repeat "N" number of times.

```
for i in range( N ):
    {for statement line1}
    {for statement line2}
    {for statement line3}
```

Use 4 spaces per indentation level

For loop from 0 to Num-1, therefore running Num times.

?

```
#The variable "i" starts out by "0" and it is incremented by "1" until it becomes 1 less than "N". #In
other words, "N" is the total amount that you want to increment up to.


a = 0
N = 10
for i in
    range(N): a =
    a + i print
    i, a


#FIY - Advanced writing
#The variable "i" starts out by being equal to "A" and it is incremented by "N" until it becomes 1 less than "B".
```

```
A = 2
B = 10
N = 2
for i in
    range(A,B,N):
    print i

```

7.1 Practice

7.1.2 Describing mathematical sequence

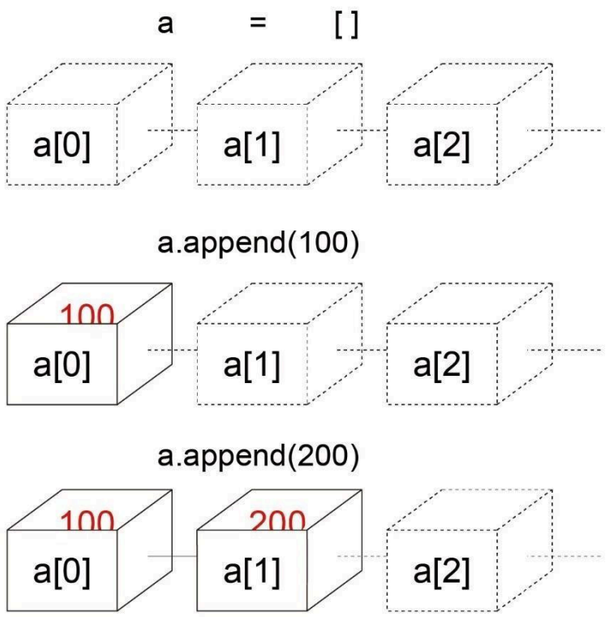
#Example:Arithmetic progression

Arithmetic progression:*https://en.wikipedia.org/wiki/Arithmetic_progression*

RHINO PYTHON LECTURE | DAY2

Contents

1. Usage of list(Array)
2. Function
3. RhinoScriptSyntax : point
4. RhinoScriptSyntax : shape
5. Installation of Grasshopper Plugin



List(Array) is variable that can store multiple values. We can declare array and pick certain value from them like following:

```
array = [0,1,2,3,4,5,6,7,8,9]
a =
array[4]
print(a)
```

List in python is variable length list and is able to be added or removed .

- .append(val) – add “val” to the last value in the list
- .remove(N) – remove “N”th value in the list

```
array = [] #blank list
print array
```

```
3       array.append(10)
4       print array
5       array.append(0)
6       print array
7       array.append('text')
8       print array
9       array.remove(0)
10      print array
```

For example, the numbers can be stored continuously in combination with For statement

```
1   array = []
2   for i in range(10):
3       array.append(i)
4   print array
```

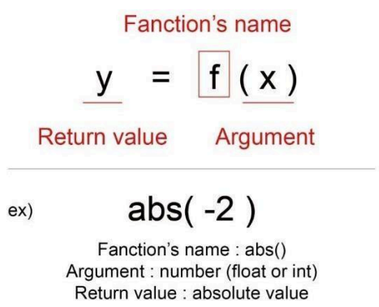
---

2. Functions

2.1. Argument / Return value

We can also call them ‘method’.

Functions are mathematical formula that have input value(=Argument) and output value(=Return value).
print() is also one of the many python’s functions (but don’t have Return value).



2.2. Built-in Functions

The Python interpreter has a number of functions built into it that are always available.

```
1
2       ##############################################################
3       #Built-in Functions
4       ##############################################################
5
6       #abs(x)
7       #Return the absolute value of a number.
8       a = abs(-2)
9       print a
10
11      #int(x)
12      #Return an integer object constructed from a number or string x, i1
13      = int(1.2030830)
14      i2 =
15      int("100")
16      print i1
17      print i2
18
19      #round(number[, ndigits])
20      #Return the floating point value number rounded to "ndigits" digits after the decimal point. #If
21      ndigits is omitted, it returns the nearest integer to its input.
22      r1 = round(1.913907) r2
23      = round(1.913907, 3)
        print r1
        print r2
```

2.3. Import Functions(Standard Library)

You can use any Python source file as a module by executing an import statement.

```
1  ############################################################
2  #Random Library
3  #Generate random numbers
4  ############################################################
5
6  #import random library
7  import random
8
9  #random.randint(a, b)
10 #Return a random integer N such that a <= N <= b. Alias for randrange(a, b+1). r
11 = random.randint(0, 10)
12 print r
13
14 #random.uniform(a, b)
15 #Return a random floating point number N such that a <= b for a <= b and b <= N <= a for b < a. u
16 = random.uniform(0.0,10.0)
17 print u
18
19 ############################################################
20 #Math Library
21 #Mathematical functions
22 ############################################################
23
24 #import math library
25 import math
26
27 #math.sqrt(x)
28 #Return the square root of x.
```

```
s =
math.sqrt(9)
print s

#math.pi
#It don't have "argument" value.
#pi - Ratio of the circumference of a circle to its diameter; Pi #The
mathematical constant pi = 3.141592..., to available precision.

p =
math.pi
print p

#math.e
#It don't have "argument" value.
#e - Napier's constant of natural logarithm
#The mathematical constant e = 2.718281..., to available precision. e
= math.e
print e


#math.sin(x)
#Return the sine of x radians.
s = math.sin(math.pi/4) #sin(45degrees)
print s

#math.cos(x)
#Return the cosine of x radians.
c = math.cos(math.pi/4) #cos(45degrees)
print c

#math.tan(x)
#Return the tangent of x radians.
t = math.tan(math.pi/4) #tan(45degrees)
print t
```

For more information, you can check — *https://docs.python.org/2.7/library/index.html*

[Tips] – import “as”

```
#############################################################
#Tips
#import "as" - Abbreviation definition of library name
#############################################################

#import and omit math library to m
import math as m
#import and omit random library to t
import random as r

r.uniform(0,10) #it can run.
print r

m.sqrt(9) #it can run.
print m
```

3. RhinoScriptSyntax : Create Points

3.1. Import RhinoScriptSyntax

To use Rhino function in python, you can use rhinoscriptsyntax. Using rhinoscriptsyntax, you need to import the library. You can import the library by following:

```
import rhinoscriptsyntax as rs
```

There are bunch of functions in rhinosctiptsyntax. Followings are examples of them.

You can draw a point, circle and polyline like following

```
import rhinoscriptsyntax as rs

#rs.AddPoint(x, y, z)
#draw a point with x,y,z coordinates
obj1 = rs.AddPoint(0,0,0)

#rs.AddCircle(center point, radius)
#draw a circle with center point and radius
obj2 = rs.AddCircle([0,0,0], 100)

#rs.AddPolyline(list of points)
#draw a polyline with a list of points
obj3 = rs.AddPolyline([[0,0,0],[100,100,0],[200,0,0]])
```

You can move and copy object like following:

```
import rhinoscriptsyntax as rs

obj = rs.AddPoint(0,0,0)

#rs.MoveObject(object_id, translation_vector)
rs.MoveObject(obj, [100,0,0])

#rs.CopyObject(object_id, copy_vector)
rs.CopyObject(obj, [0,100,0])
```

9

You can get object from Rhino like following:

```
1    import rhinoscriptsyntax as rs
2
3    #rs.GetObject() - don't need argument
4    obj = rs.GetObject()
5
6    rs.CopyObject(obj, [0,100,0])
```

For more information, you can check at the menu bar of RhinoPython Script Editor

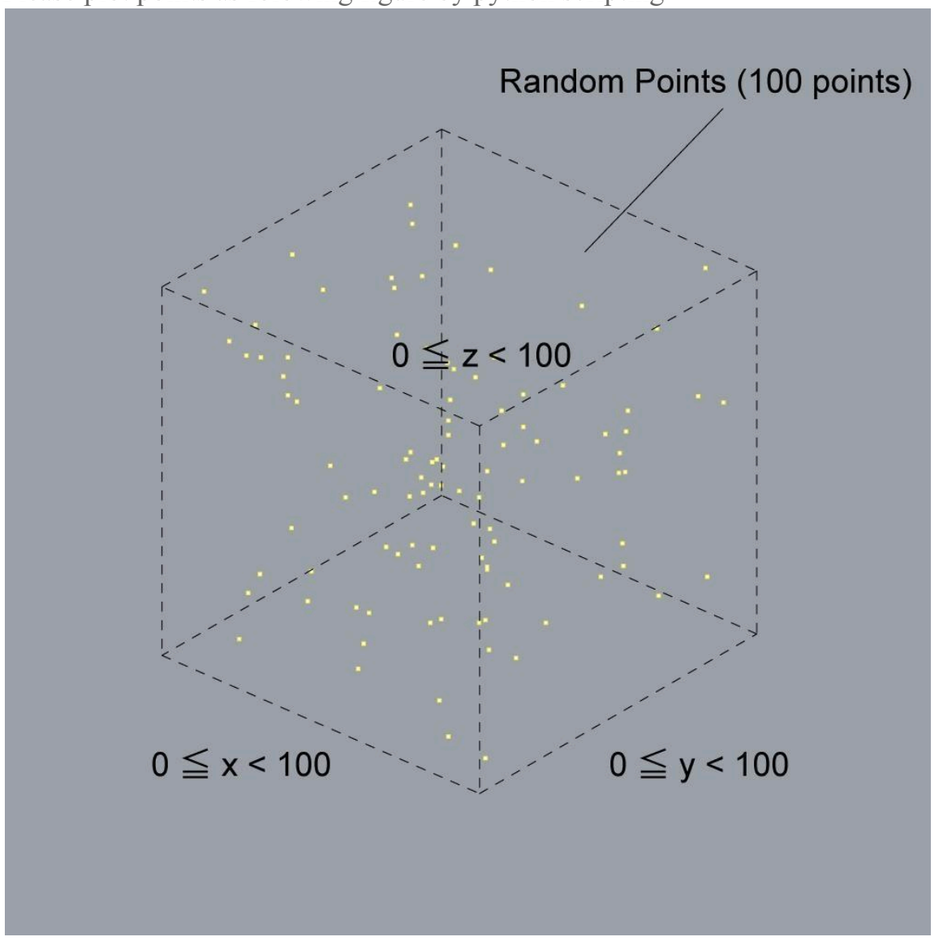
```
Help -> Pyhton Help
```

Or check — *http://developer.rhino3d.com/api/RhinoScriptSyntax/win/*

Practice

A. Random Points

Please plot points as folowing figure by python scripting



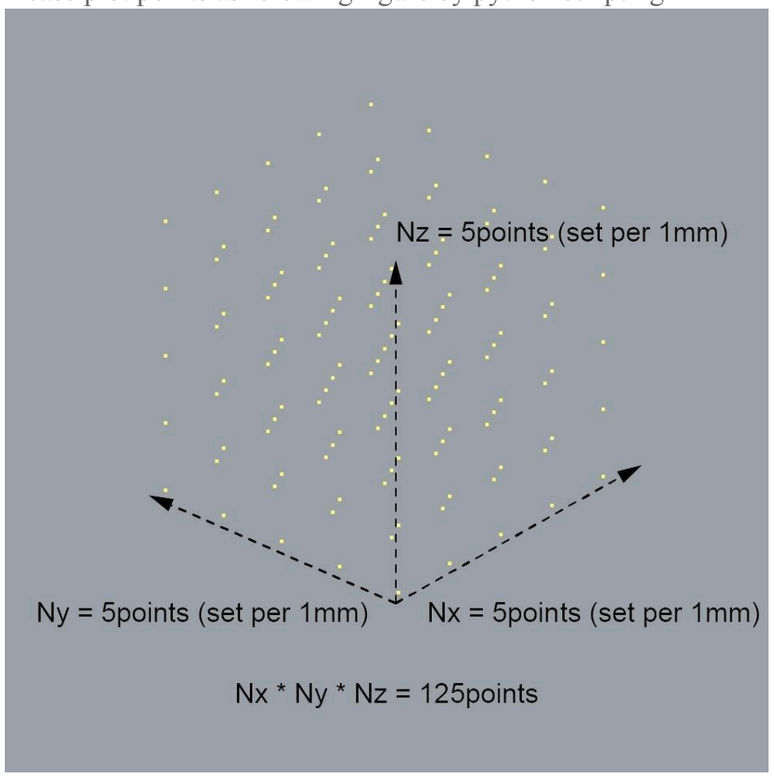
Hint

```
import rhinoscriptsyntax as rs
import random as rnd
```

```
3    rs.AddPoint(rnd.uniform(0,100),rnd.uniform(0,100),rnd.uniform(0,100))
4
```

B. Matrix of Points

Please plot points as folowing figure by python scripting



Hint

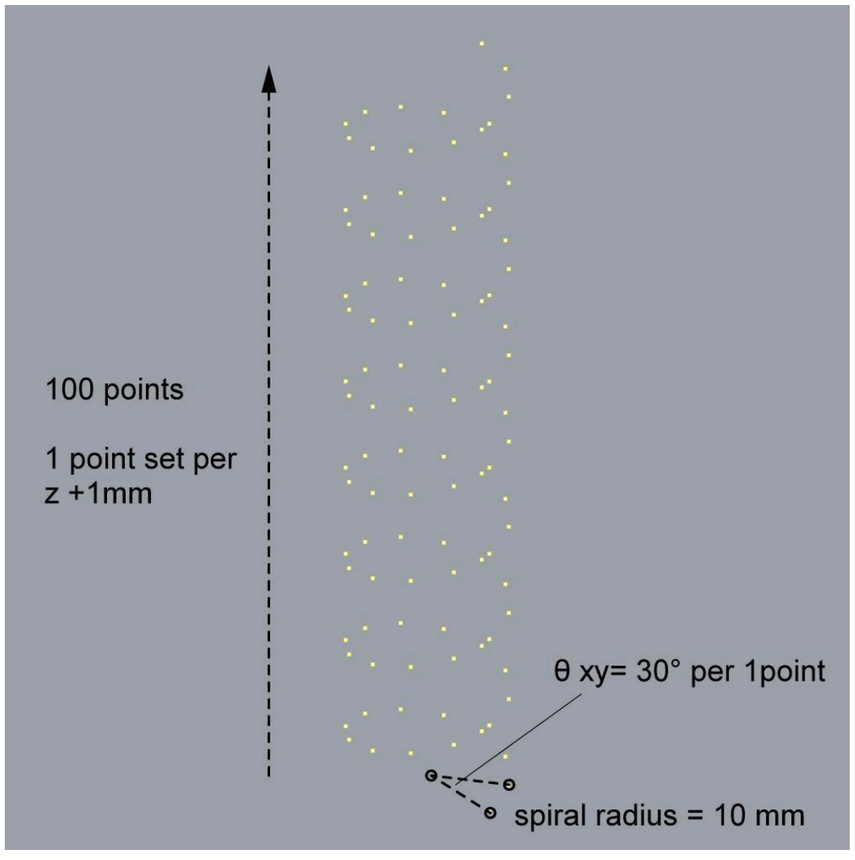
[?]

```
1    for i in range(5):
         for j in range(5):
```

```
2               print i, j
3
```

C. Spiral Points Pattern

Please plot points as folowing figure by python scripting
[reference]Trigonometric functions:*https://en.wikipedia.org/wiki/Trigonometric_functions*



Hint

```
?
1     for i in range(6):
2         rs.AddPoint([10*math.cos(math.pi/6*i), 10*math.sin(math.pi/6*i),0] )
```

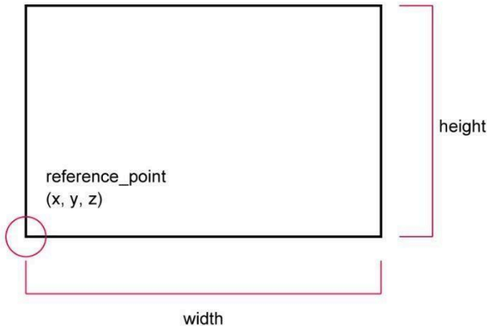
---

4. RhinoScriptSyntax : Create Models

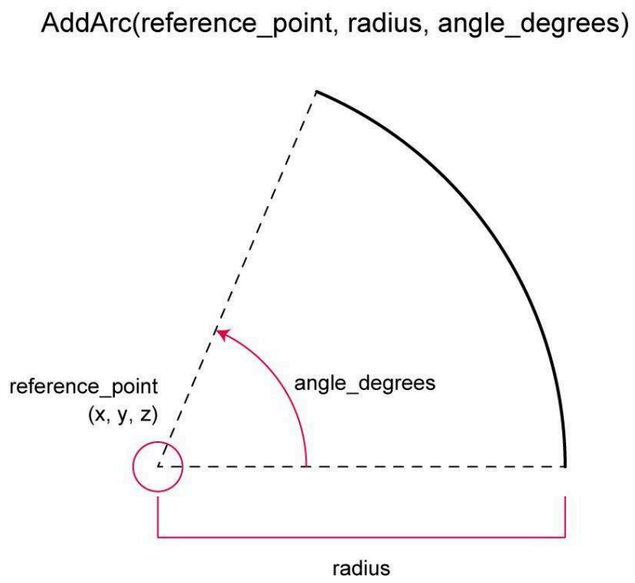
4.1. Draw Models

Ex1)AddRectangle()

AddRectangle(reference_point, width, height)



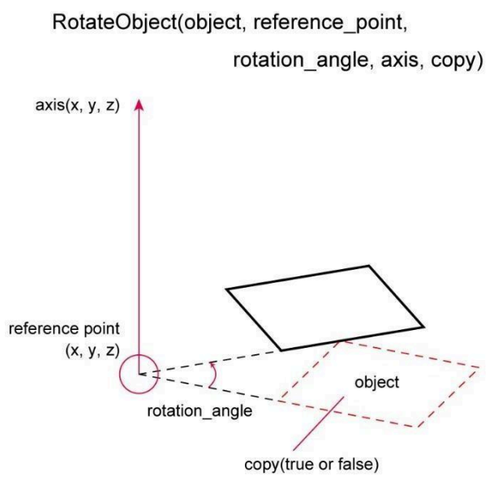
Ex2) AddArc()




```
import rhinoscriptsyntax as rs


r = 5.0
origin = [0.0, 0.0, 0.0]

rec = rs.AddRectangle(origin, r, r)
arc = rs.AddArc(origin, r, 90.0)

```

4.2. Edit Models

Ex3)RotateObject()



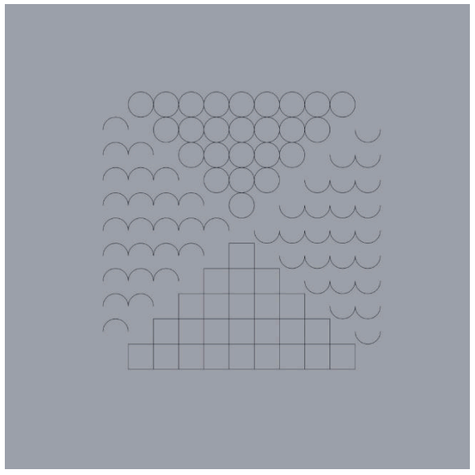
[?](#)

```
import rhinoscriptsyntax as rs
r = 5.0
origin = [0.0, 0.0, 0.0]

rec = rs.AddRectangle(origin, r, r)

for i in range(16):
    rs.RotateObject(rec, origin, 30.0*i, [0,0,1], True)
```

#Practice

A. Pattern of diagonal area

Please draw Pattern of diagonal area as folowing figure by python scripting



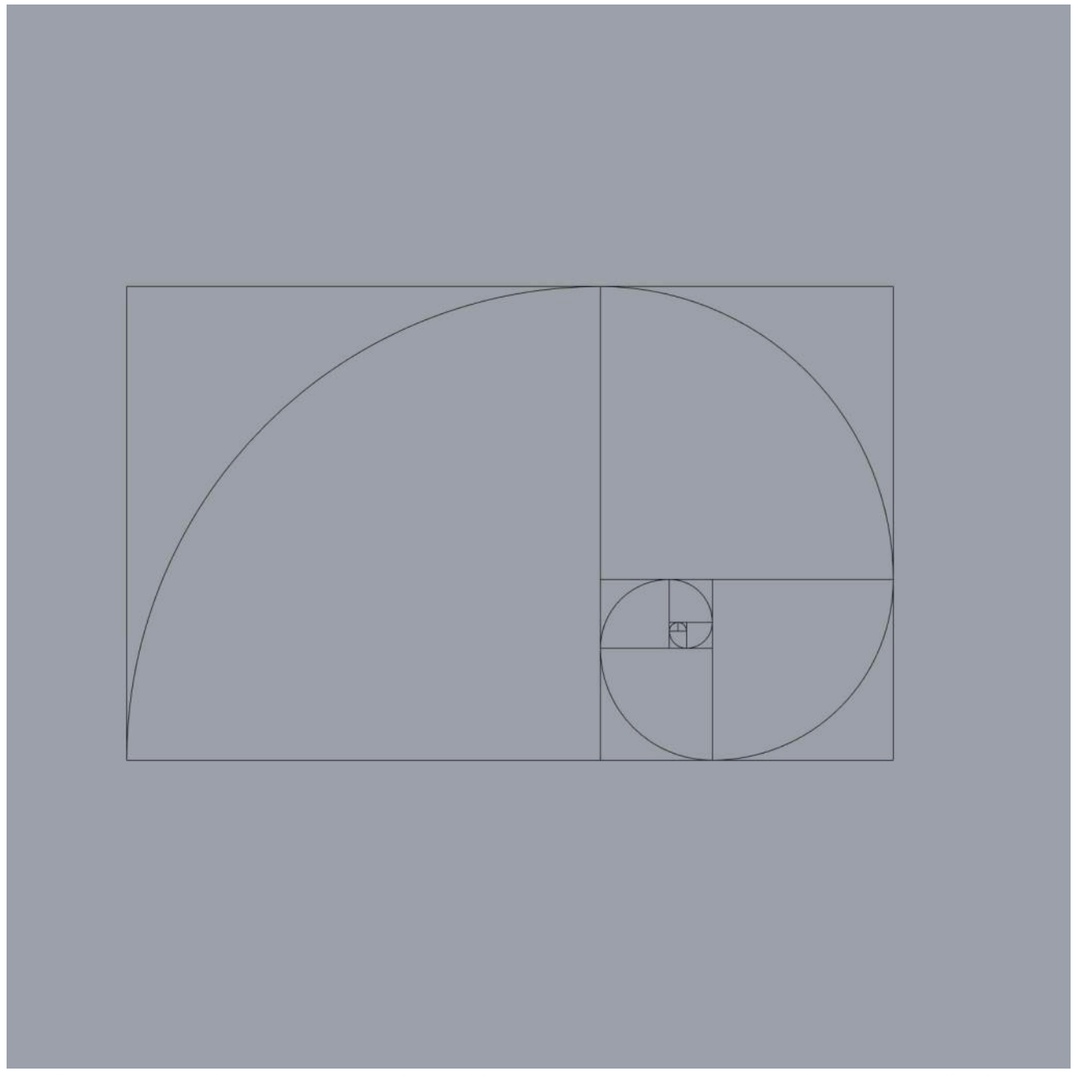
https://en.wikipedia.org/wiki/Inequality_(mathematics)

Hint

```
import rhinoscriptsyntax as rs


n = 10


for x in range(n+1):
    for y in range(n+1):
        if(y &amp;amp;amp;amp;amp;amp;amp;amp;gt; x and y &amp;amp;amp;amp;amp;amp;amp;amp;gt; -x + n):
            rs.AddCircle([x, y, 0.0], 0.5)

```

B. The golden spiral

Please draw the pattern of “The golden spiral” as folowing figure by python scripting



A Fibonacci spiral approximates the golden spiral using quarter-circle arcs inscribed in squares of integer Fibonacci-number side, shown for square sizes 1, 1, 2, 3, 5, 8, 13, 21, and 34.

https://en.wikipedia.org/wiki/Golden_spiral
https://en.wikipedia.org/wiki/Fibonacci_number

Hint

```
import rhinoscriptsyntax as rs

roopNum = 10
originPt = [0.0, 0.0, 0.0]

for i in range(roopNum):

    if(i == 0):
        fn1 = 0 #the initial term of Fibonacci number
        fn = 1 #the first term of Fibonacci number
    else:
        fn = fn1 + fn0 #Fibonacci progression

    #Print Fibonacci Num
    print fn

    #Drawing Recrangle
    rec = rs.AddRectangle(originPt, fn, fn)
    rec = rs.RotateObject(rec, originPt, 90.0*i)

    #Drawing Arc
    arc = rs.AddArc(originPt, fn, 90.0)
    arc = rs.RotateObject(arc, originPt, 90.0*i)

    fn0 = fn1 #Replacement of data for next step (f0)
    fn1 = fn #Replacement of data for next step (f1)
```

Answer1: change origin point

```
import rhinoscriptsyntax as rs
import Rhino as rh

roopNum = 10

for i in range(roopNum):

    if(i == 0):
        fn1 = 0 #the initial term of Fibonacci number
        fn = 1 #the first term of Fibonacci number
        originPt = [0.0, 0.0, 0.0]
    else:
        fn = fn1 + fn0 #Fibonacci progression
        if i % 4 == 0:
            originPt = [originPt[0] - fn0, originPt[1], originPt[2]]
        elif i % 4 == 1:
            originPt = [originPt[0], originPt[1] - fn0, originPt[2]]
        elif i % 4 == 2:
            originPt = [originPt[0] + fn0, originPt[1], originPt[2]]
        else:
            originPt = [originPt[0], originPt[1] + fn0, originPt[2]]

    #Print Fibonacci Num
    print fn



    #Drawing Recrangle
    rec = rs.AddRectangle(originPt, fn, fn)
    rec = rs.RotateObject(rec, originPt, 90.0*i)
```

```
        #Drawing Arc
        arc = rs.AddArc(originPt, fn, 90.0)
        arc = rs.RotateObject(arc, originPt, 90.0*i)

        fn0 = fn1 #Replacement of data for next step (f0)
        fn1 = fn #Replacement of data for next step (f1)
```

Answer2: move

```
import rhinoscriptsyntax as rs
import Rhino as rh

roopNum = 10
originPt = [0.0, 0.0, 0.0]

for i in range(roopNum):

    if(i == 0):
        fn1 = 0 #the initial term of Fibonacci number
        fn = 1 #the first term of Fibonacci number
        movePt = [0.0, 0.0, 0.0]
    else:
        fn = fn1 + fn0 #Fibonacci progression
        if i % 4 == 0:
            movePt = [movePt[0] - fn0, movePt[1], movePt[2]]
        elif i % 4 == 1:
            movePt = [movePt[0], movePt[1] - fn0, movePt[2]]
        elif i % 4 == 2:
            movePt = [movePt[0] + fn0, movePt[1], movePt[2]]
```

```
        else:
            movePt = [movePt[0], movePt[1] + fn0, movePt[2]]

    #Print Fibonacci Num
    print fn



    #Drawing Recrangle
    rec = rs.AddRectangle(originPt, fn, fn)
    rec = rs.RotateObject(rec, originPt, 90.0*i)
    rec = rs.MoveObject(rec, movePt)

    #Drawing Arc
    arc = rs.AddArc(originPt, fn, 90.0)
    arc = rs.RotateObject(arc, originPt, 90.0*i)
    arc = rs.MoveObject(arc, movePt)

        fn0 = fn1 #Replacement of data for next step (f0)
        fn1 = fn #Replacement of data for next step (f1)
```

Answer3: math

?

```
import rhinoscriptsyntax as rs
import Rhino as rh
import math

roopNum = 10

for i in range(roopNum):
```

```
    if(i == 0):
        fn1 = 0 #the initial term of Fibonacci number
        fn = 1 #the first term of Fibonacci number
        originPt = [0.0, 0.0, 0.0]
    else:
        fn = fn1 + fn0 #Fibonacci progression
        originPt = [originPt[0] - (fn0 * math.cos(math.pi * i / 2)), originPt[1] - (fn0 * math.sin(math.pi * i / 2)),
        originPt[2]]

    #Print Fibonacci Num
    print fn



    #Drawing Recrangle
    rec = rs.AddRectangle(originPt, fn, fn)
    rec = rs.RotateObject(rec, originPt, 90.0*i)

    #Drawing Arc
    arc = rs.AddArc(originPt, fn, 90.0)
    arc = rs.RotateObject(arc, originPt, 90.0*i)

    fn0 = fn1 #Replacement of data for next step (f0)
    fn1 = fn #Replacement of data for next step (f1)
```

5. Installation of Grasshopper Plugin

Get GHPython from following Site, if you use Rhino5.

https://www.food4rhino.com/app/ghpython

You can get various plugin here.

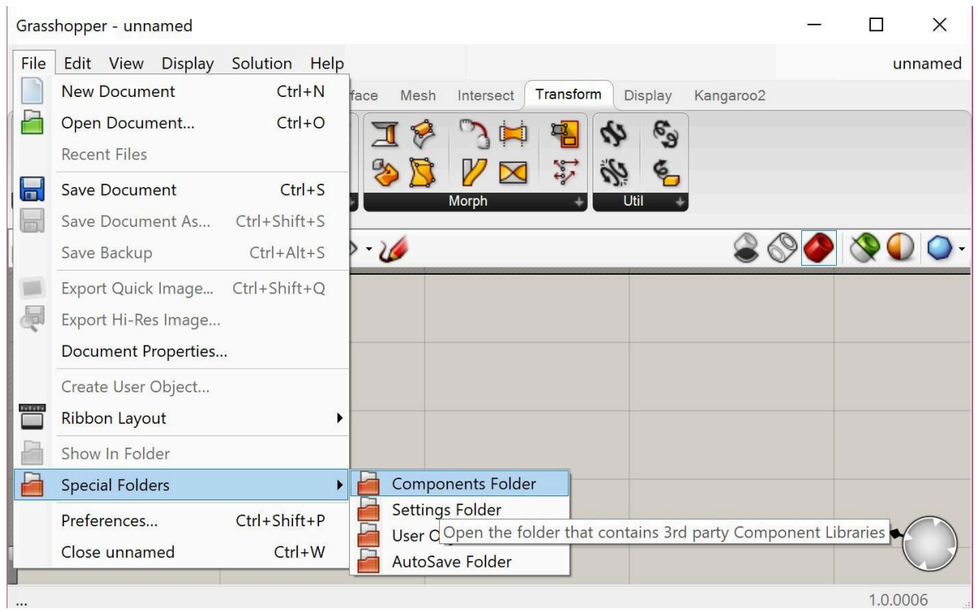
Recommendation is

- LUNCHBOX : *https://www.food4rhino.com/app/lunchbox*
  - plugin for exploring mathematical shapes, paneling, structures, and workflow.
- KANGAROO PHYSICS : *https://www.food4rhino.com/app/kangaroo-physics*
  - plugin for physical simulation.
- LADYBUG TOOLS : *https://www.food4rhino.com/app/ladybug-tools#downloads_list*
  - plugin for environment simulation.

If you does get not an installer but a zip file from the download page, you have to install them manually.

1. Unzip the download file.

2. Click “File” – “Special Folder” – “Component Folder” to open the component folder, which has plugin data.



3. Copy unziped files into components folder.

4. Restart Rhinoceros.

5.2. UI

1. Set GHPython Component

2. GHPython Editer
    - Start-up Ghpython Editer

        — Double Click Ghpython Component

- Compare Python Editer and GHPython Editer

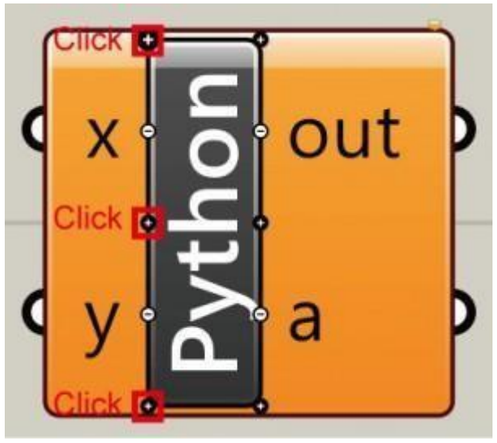
RHINO PYTHON LECTURE | DAY3

Contents

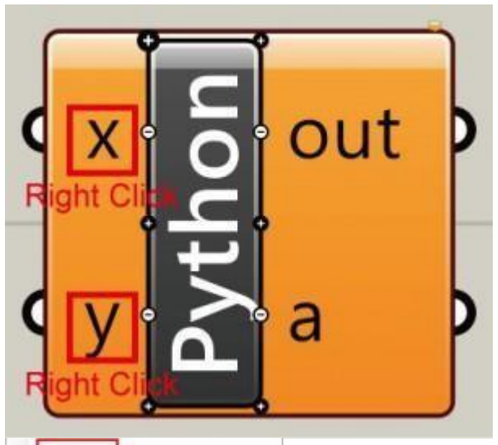
1. Practice
    1. Parametric spark model
    2. The Golden Spiral – convert Python code to Ghpython code

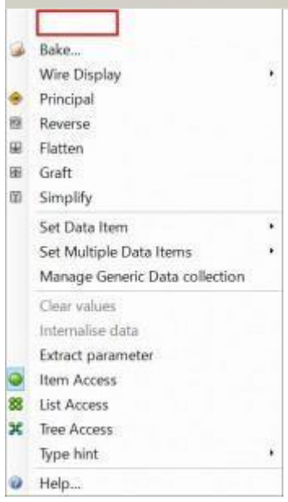
1. GHPython – Input Parameter
    1. Input variable declaration
    2. Access
    3. Type hint
2. GHPython – Output Parameter
    1. Output variable declaration
    2. “Out” value – Console
3. Rhinoscriptsyntax in GH

1. GHPython – Input Parameter
    - 2.1. Input variable declaration
    - Increase variable

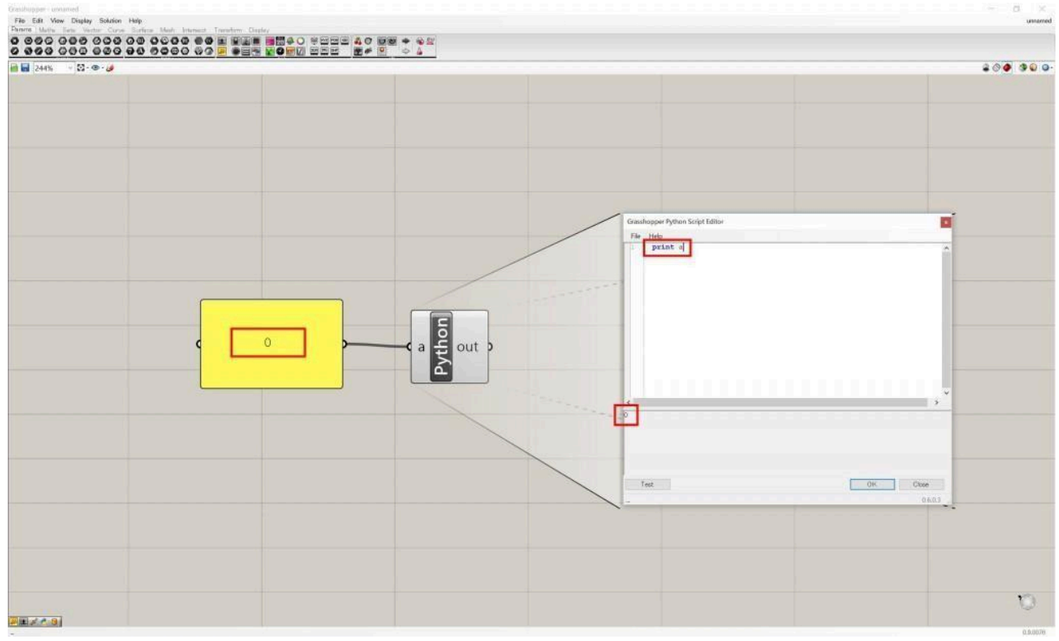


- Name(Rename) variable



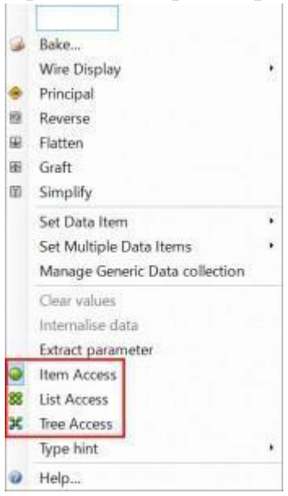


- Check input



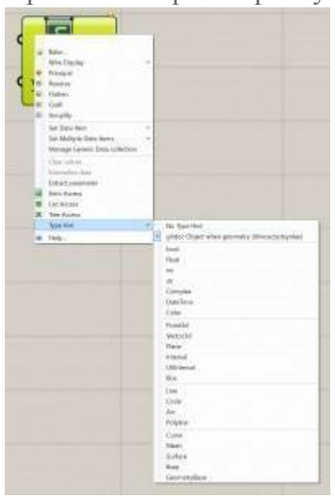
- 2.2. Access

Input value requires specifying “data Access type”



- Item Access — Python or GH’s single value
- List Access — Python’s List
- Tree Access — GH’s DataTree
- 2.3. Type hint

Input value requires specifying “Type hint”

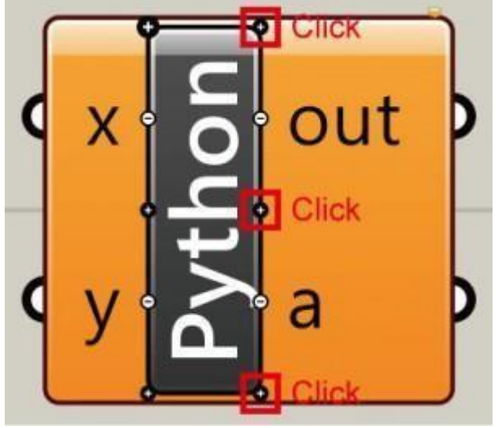


---

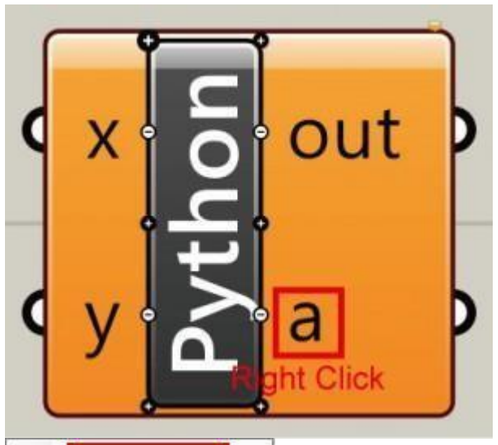
2. GHPython – Output Parameter

2.1. Output variable declaration

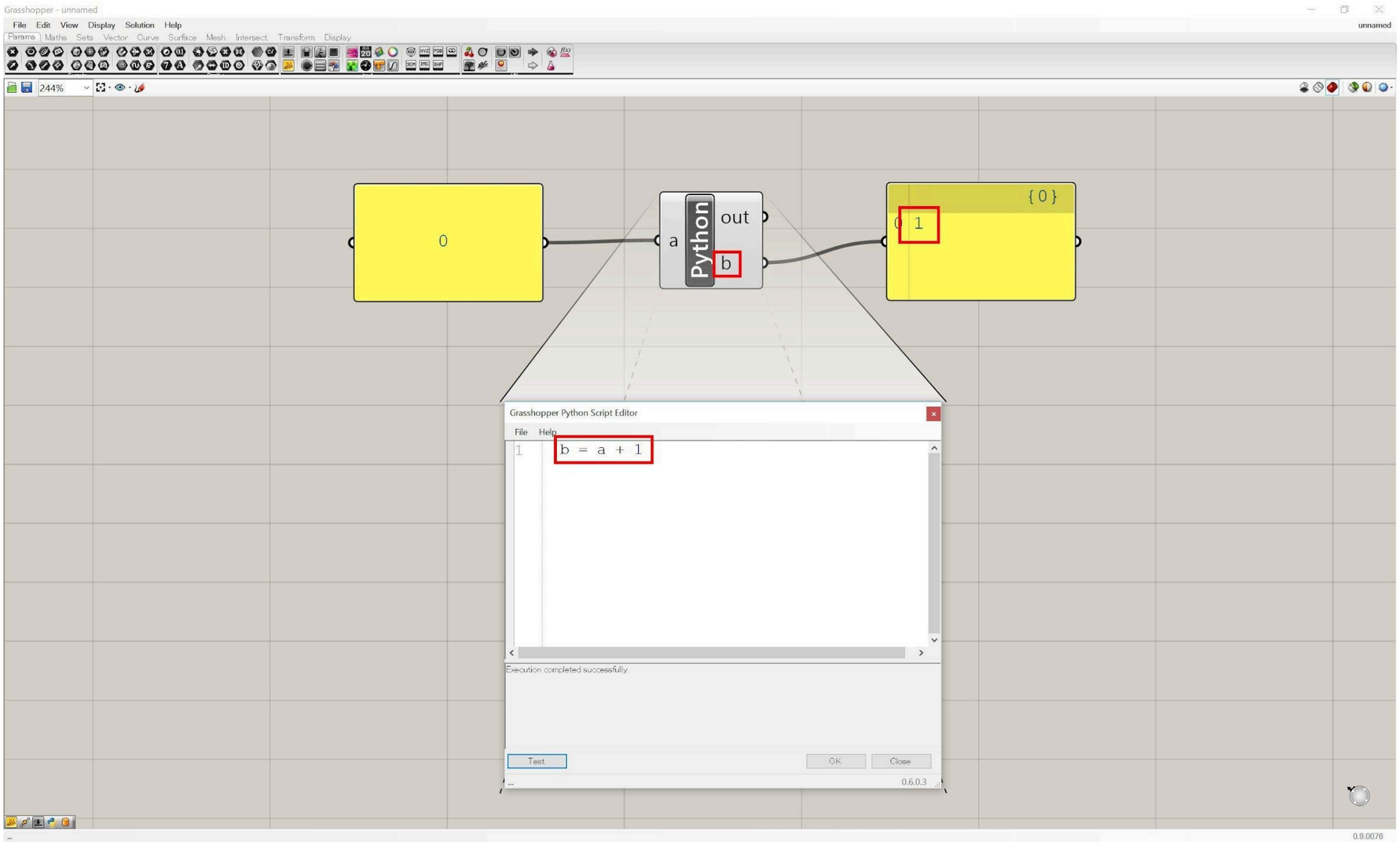
- Increase variable



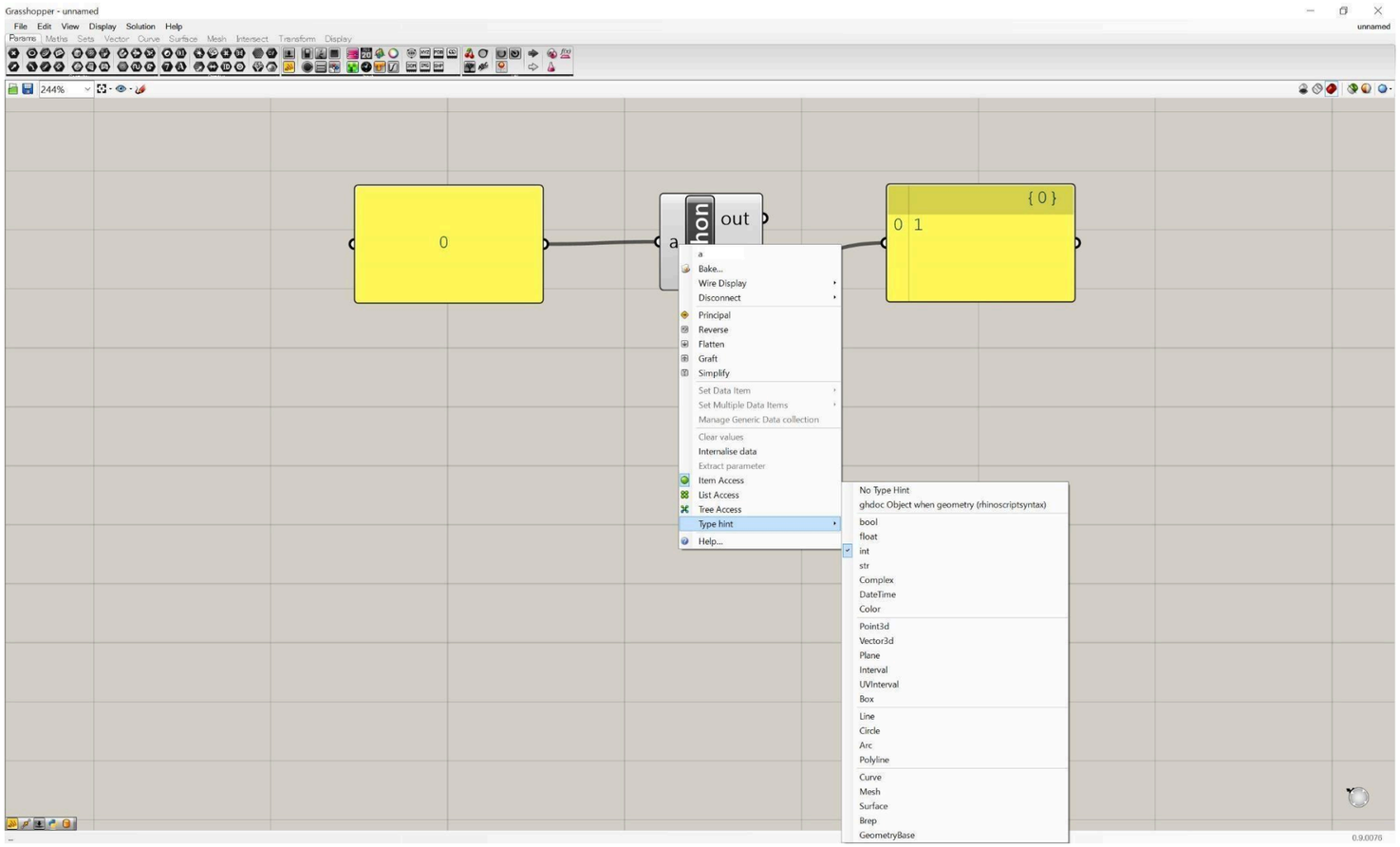
- Name(Rename) variable



- Check output

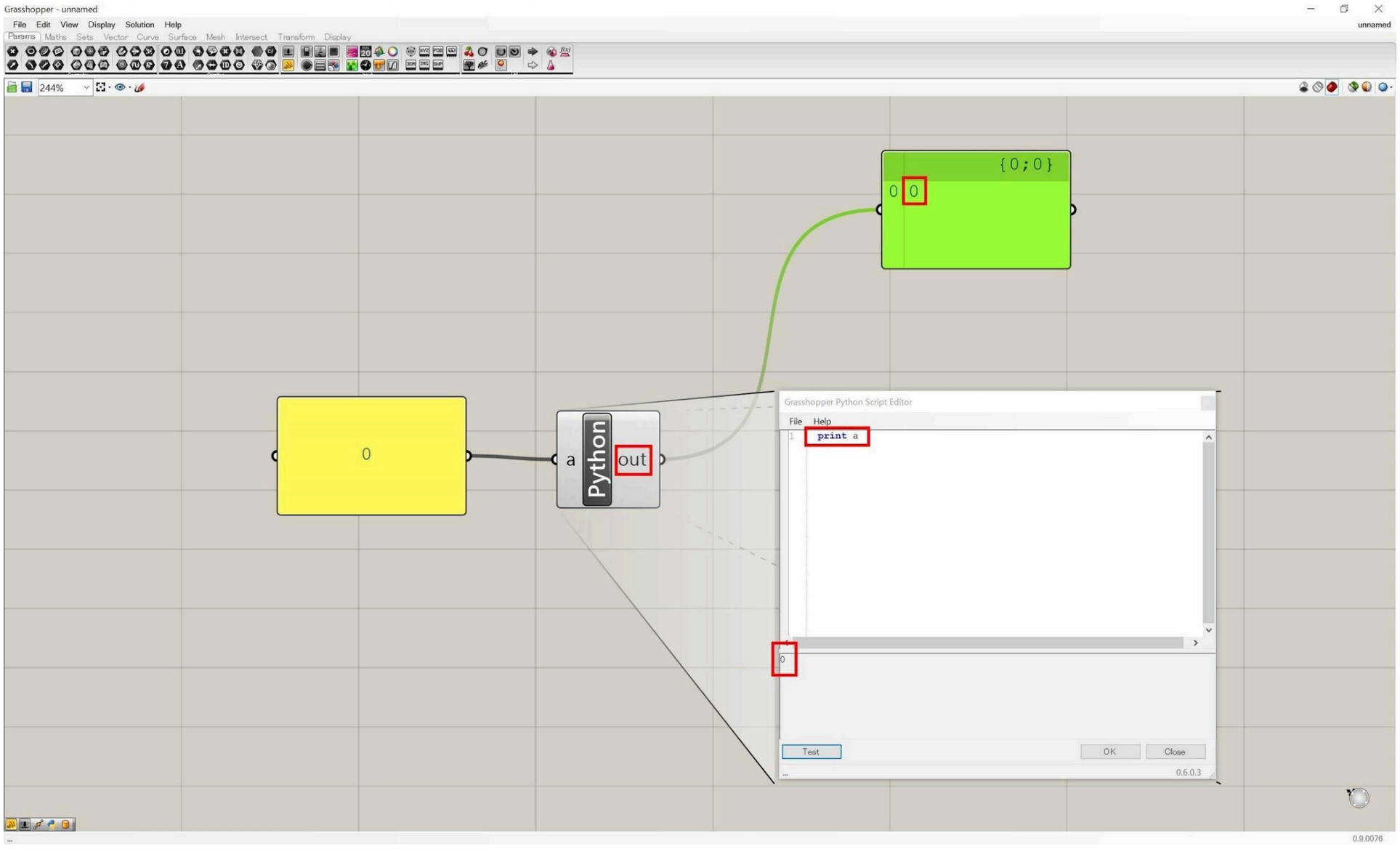


Notice: Type hint of value “a” = **int**

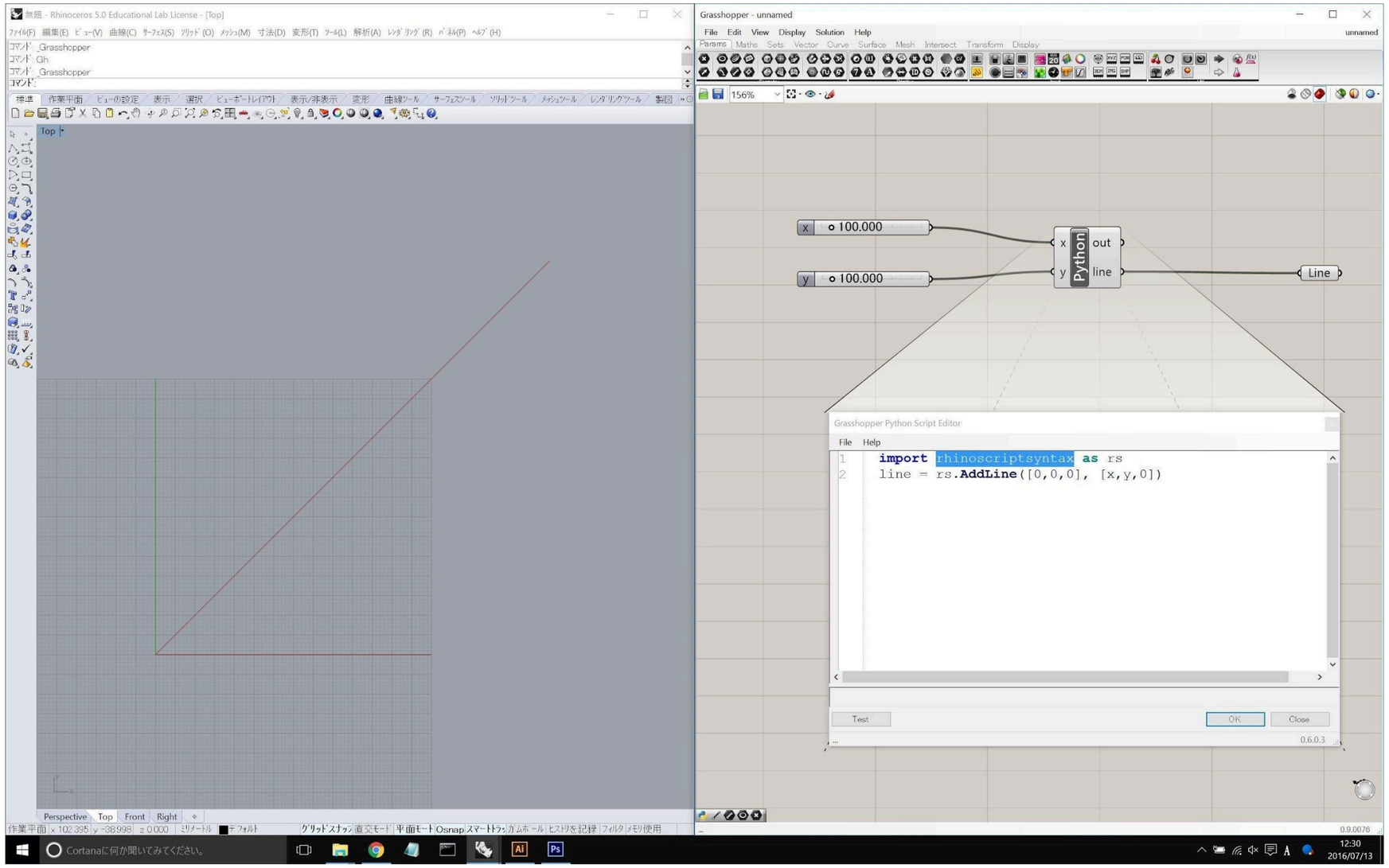


2.2. “Out” value – Console

“Out” value is preset output value. Connecting to Pannel Component, It show cosole data.



3. Rhinoscriptsyntax in GH

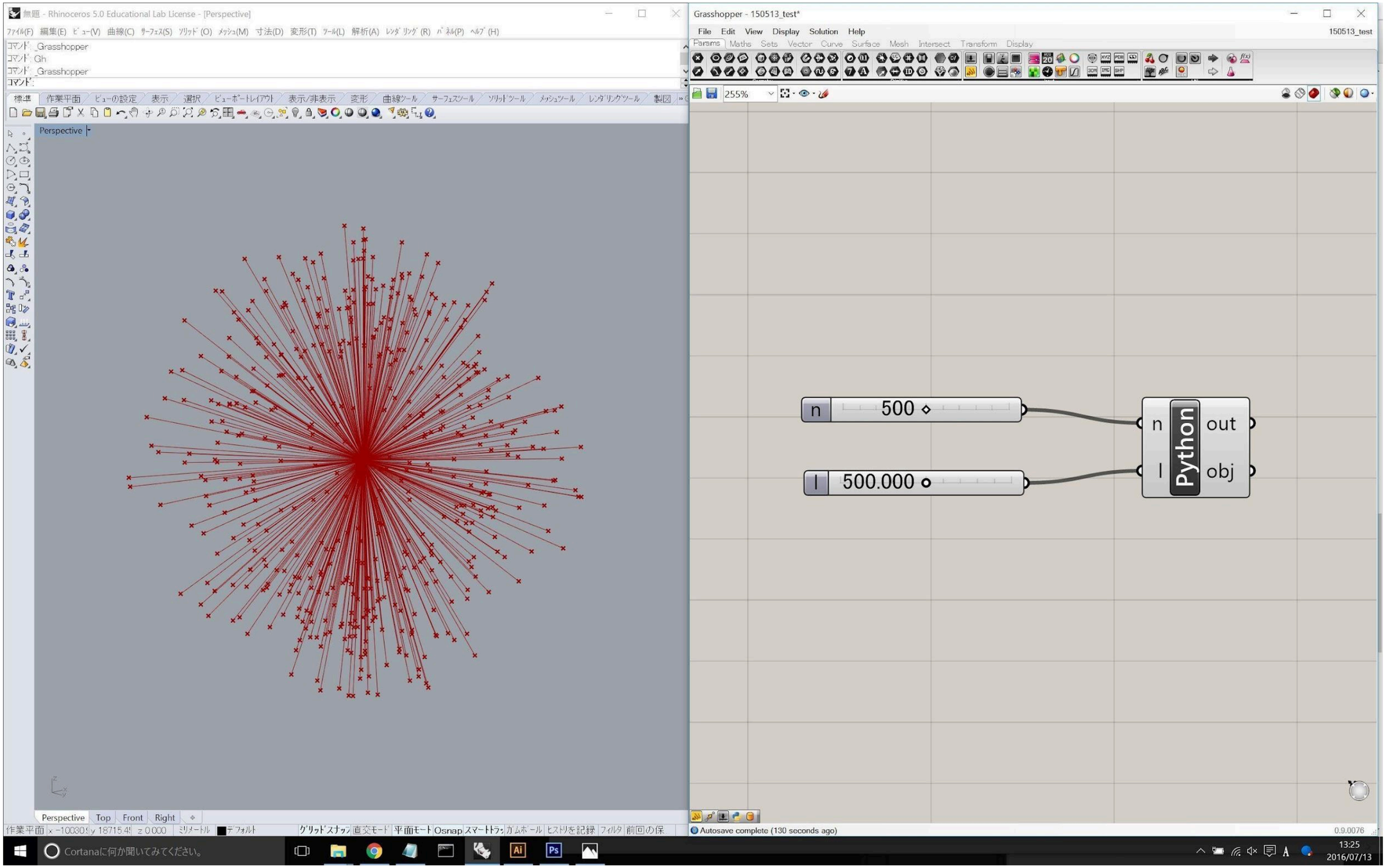


---

4. Practice

4.1 Parametric spark model

Please draw Parametric spark model as folowing figure by GHpython scripting



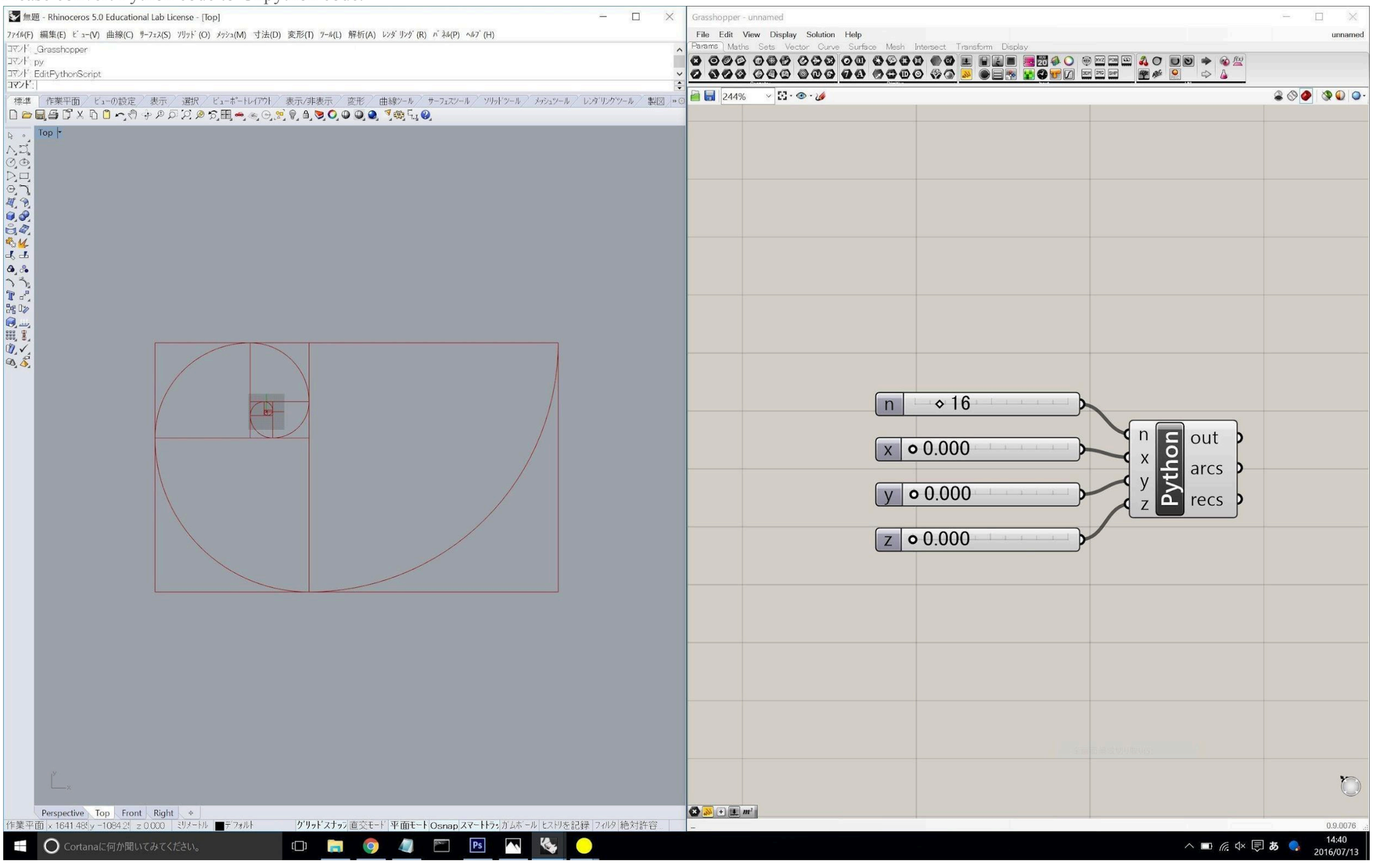
- Random value
  - Lengths of spark’s lines

    - Angles of spark’s lines
- Input value
    - input value1 “n”:Number of spark’s lines
    - input value2 “l”:Max length of spark’s lines

Hint

4.2 The Golden Spiral

Please convert Python code to Ghpython code.



- Input value
  - input value “n”:Number of Fibonacci terms
  - input value “x”:X value of start point
  - input value “y”:Y value of start point
  - input value “z”:Z value of start point

- Algorithm of drawing The Golden Spiral

?

```
import rhinoscriptsyntax as rs


roopNum = 10
originPt = [0.0, 0.0, 0.0]


for i in range(roopNum):


    if i == 0 :
        fn1 = 0 #the initial term of Fibonacci number
        fn = 1 #the first term of Fibonacci number
    else:
        fn = fn1 + fn0 #Fibonacci progression


    #Print Fibonacci Num
    print fn


    #Replacement of data for originPt
    if not i == 0:
        if i%4 == 0:
            originPt[0] -=
        fn0 elif i%4 == 1:
            originPt[1] -=
        fn0 elif i%4 == 2:
            originPt[0] += fn0
```

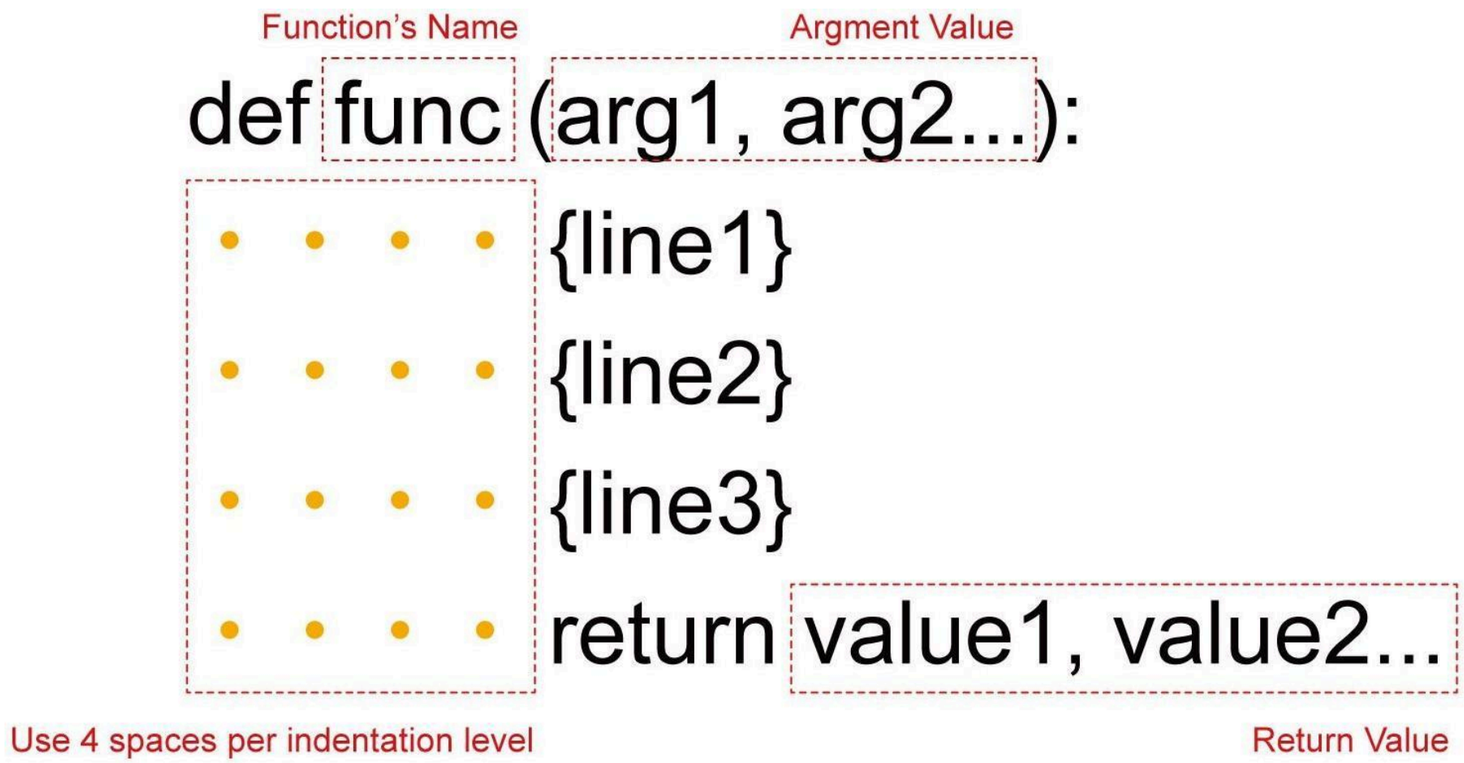
```
        elif i%4 == 3:
            originPt[1] += fn0

    #Drawing Recrangle
    rec = rs.AddRectangle(originPt, fn, fn)
    rec = rs.RotateObject(rec, originPt, 90.0*i)

    #Drawing Arc
    arc = rs.AddArc(originPt, fn, 90.0)
    arc = rs.RotateObject(arc, originPt, 90.0*i)

    fn0 = fn1 #Replacement of data for next step (f0)
    fn1 = fn #Replacement of data for next step (f1)
```

RHINO PYTHON LECTURE | DAY4

Contents

1. User-defined function

We can define and edit original function in python. It is called User-defined functions. Defining it, you use same algorithm repeatly.We can edit also number and type of argument and return values.



?

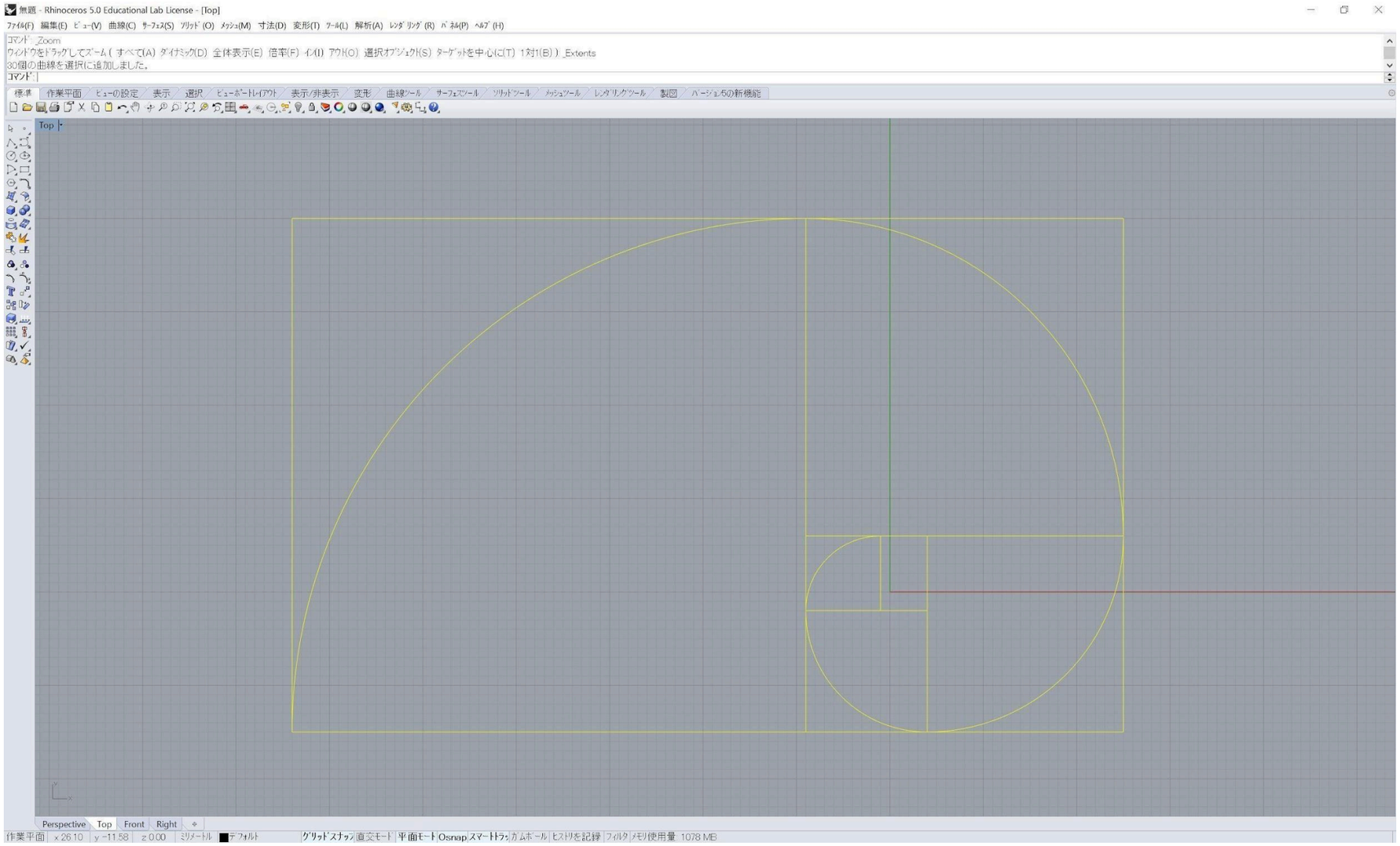
```
1    import rhinoscriptsyntax as rs
```

```
2
3   def UserDefFunc1(a, b):
4
5       #Add two values
6       c = a + b
7
8       #return as single value
9       return c
10
11  #Call UserDefFunc1 and Store return value
12  f1 = UserDefFunc1(5, 6)
13  #Print value
14  print f1
15
16  def UserDefFunc2(a, b):
17      #Add two values
18      c = a + b
19
20      #Subtract two values
21      d = a - b
22
23      #return as array
24      return c, d
25
26  #Call UserDefFunc2 and Store return value
27  f2 = UserDefFunc2(5, 6)
28  #Print value
29  print
30  f2[0]
    print
    f2[1]
```

#Practice

A. The Golden Spiral – User-defined functions

Please convert Python code to User-defined functions.



- Argument value
  - First Number of Fibonacci

        - Loop Number
        - X value of start point
        - Y value of start point
        - Z value of start point
    - Return value
        - Rectangle objects
        - Arc objects
- Algorithm of drawing The Golden Spiral

[?]

```
import rhinoscriptsyntax as rs

loopNum = 10
originPt = [0.0, 0.0, 0.0]

for i in
    range(loopNum): if
    i == 0 :
        fn1 = 0 #the initial term of Fibonacci number
        fn = 1 #the first term of Fibonacci number
    else:
        fn = fn1 + fn0 #Fibonacci progression

    #Print Fibonacci Num
    print fn

    #Replacement of data for originPt
    if not i == 0:
        if i%4 == 0:
            originPt[0] -=
        fn0 elif i%4 == 1:
            originPt[1] -=
        fn0 elif i%4 == 2:
            originPt[0] += fn0
```

```
elif i%4 == 3:
```

```
23                 originPt[1] += fn0
24
25         #Drawing Recrangle
26         rec = rs.AddRectangle(originPt, fn, fn)
27         rec = rs.RotateObject(rec, originPt, 90.0*i)
28
29         #Drawing Arc
30         arc = rs.AddArc(originPt, fn, 90.0)
31         arc = rs.RotateObject(arc, originPt, 90.0*i)
32
33         fn0 = fn1 #Replacement of data for next step (f0)
34         fn1 = fn #Replacement of data for next step (f1)
35
36
```

2. paradigms

2.1 Procedural programming

reference: *https://en.wikipedia.org/wiki/Procedural_programming*

In this paradigm, we use routine which is main process, subroutine which is branched process divided from routine, and function which return result after calculate something.

We don’t have explicit sets of data in the process.

It’s suitable for Fortran, ALGOL, COBOL and BASIC, they are little bit old languages.

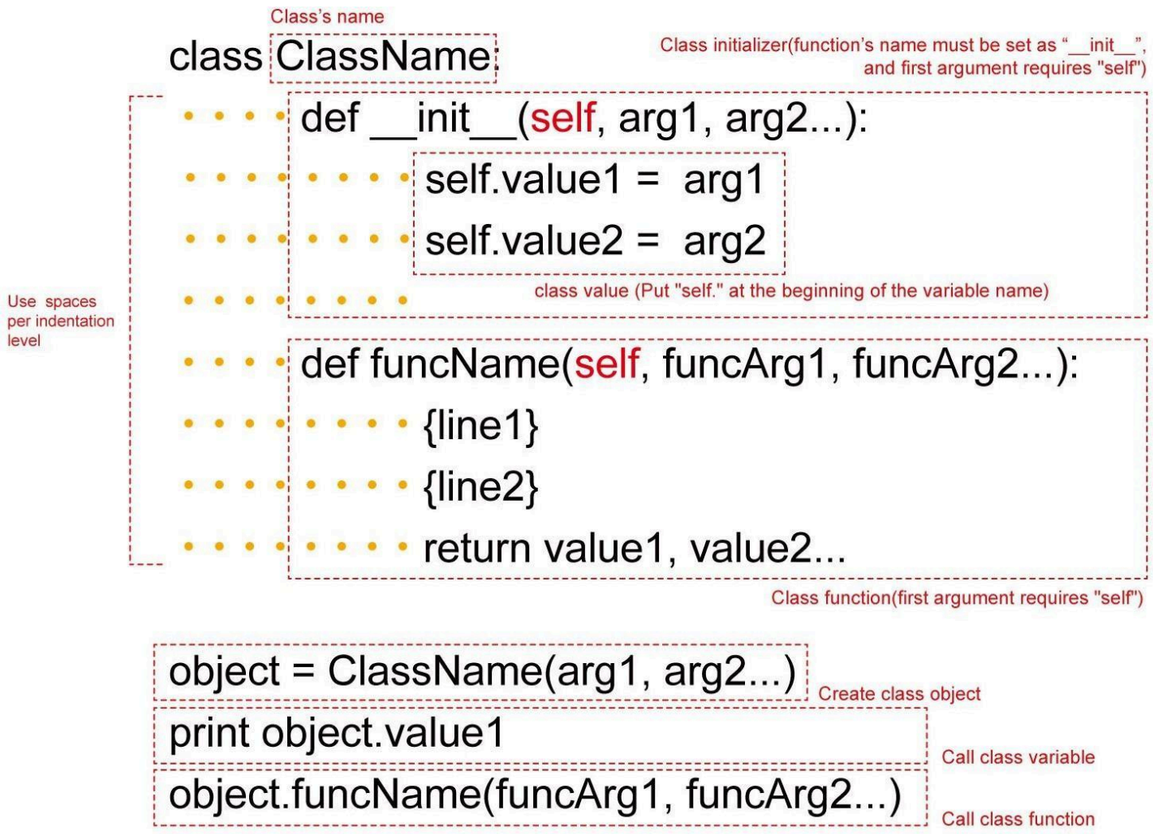
2.2 Object-oriented programming

Object-oriented programming (OOP) is a programming paradigm based on the concept of “objects”, which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. A feature of objects is that an object’s procedures can access and often modify the data fields of the object with which they are associated (objects have a notion of “this” or “self”). In OOP, computer programs are designed by making them out of objects that interact with one another. There is significant diversity of OOP languages, but the most popular ones are class-based, meaning that objects are instances of classes, which typically also determine their type.

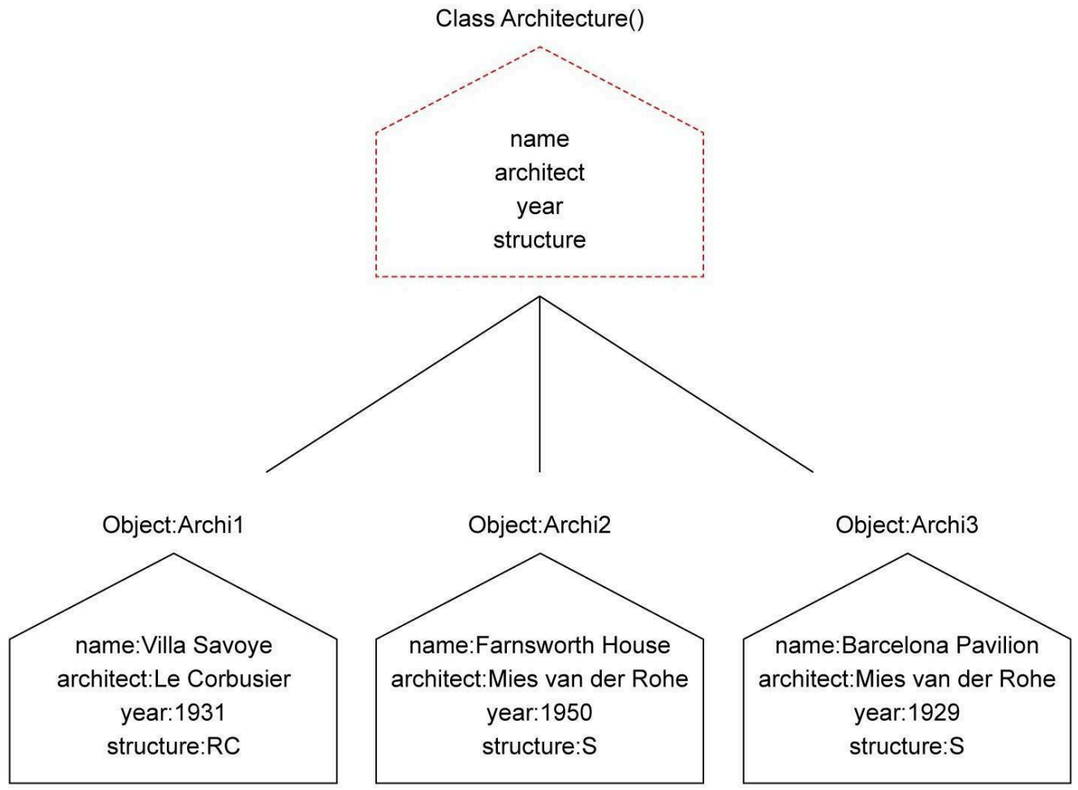
Many of the most widely used programming languages are multi-paradigm programming languages that support object-oriented programming to a greater or lesser degree, typically in combination with imperative, procedural programming.

In a simple term, we use object which have set of data and procedure for programming.

‘Class’ is a something like blueprints. we generate object called ‘Instance’ from ‘Class’ in the program.



ex)Architecture() Class



Architecture Class have 4 variables and 2 functions.If we call them, objects return self data at any time. So They behave in bottom-up, not in top-down.

?

```
class Architecture:


    #Class initializer
    def_init_(self, _name, _architect, _year, _structure):
        #Class variable
        self.name = _name
        self.architect = _architect
        self.year = _year
        self.structure = _structure
```

```
    #Class function 1
    def ageDifference(self, yourArchi):
        ageDif = self.year - yourArchi.year
        if(ageDif > 0):
            print self.name, "was built after", ageDif, "year(s) after", yourArchi.name, "was."
        elif(ageDif < 0):
            print self.name, "was built after", -ageDif, "year(s) before", yourArchi.name, "was."
        else:
            print self.name, "and", yourArchi.name , "are built in same year."

    #Class function 2
    def compareArchitect(self, yourArchi):
        if(self.architect == yourArchi.architect):
            print "The architects of", self.name, "and", yourArchi.name, "are same."
        else:
            print "The architects of", self.name, "and", yourArchi.name, "are not same."


#Create "Architecture" class object

archi2 = Architecture("Farnsworth House", "Mies van der Rohe", 1950, "S") archi3
= Architecture("Barcelona Pavilion", "Mies van der Rohe", 1929, "S")

#Call "Architecture" class variable
print archi1.name
print archi2.year
print archi3.structure

#Call "Architecture" class functions
archi1.ageDifference(archi2)
archi2.compareArchitect(archi1)
archi3.compareArchitect(archi3)
```

38

39

40

41

#Practice

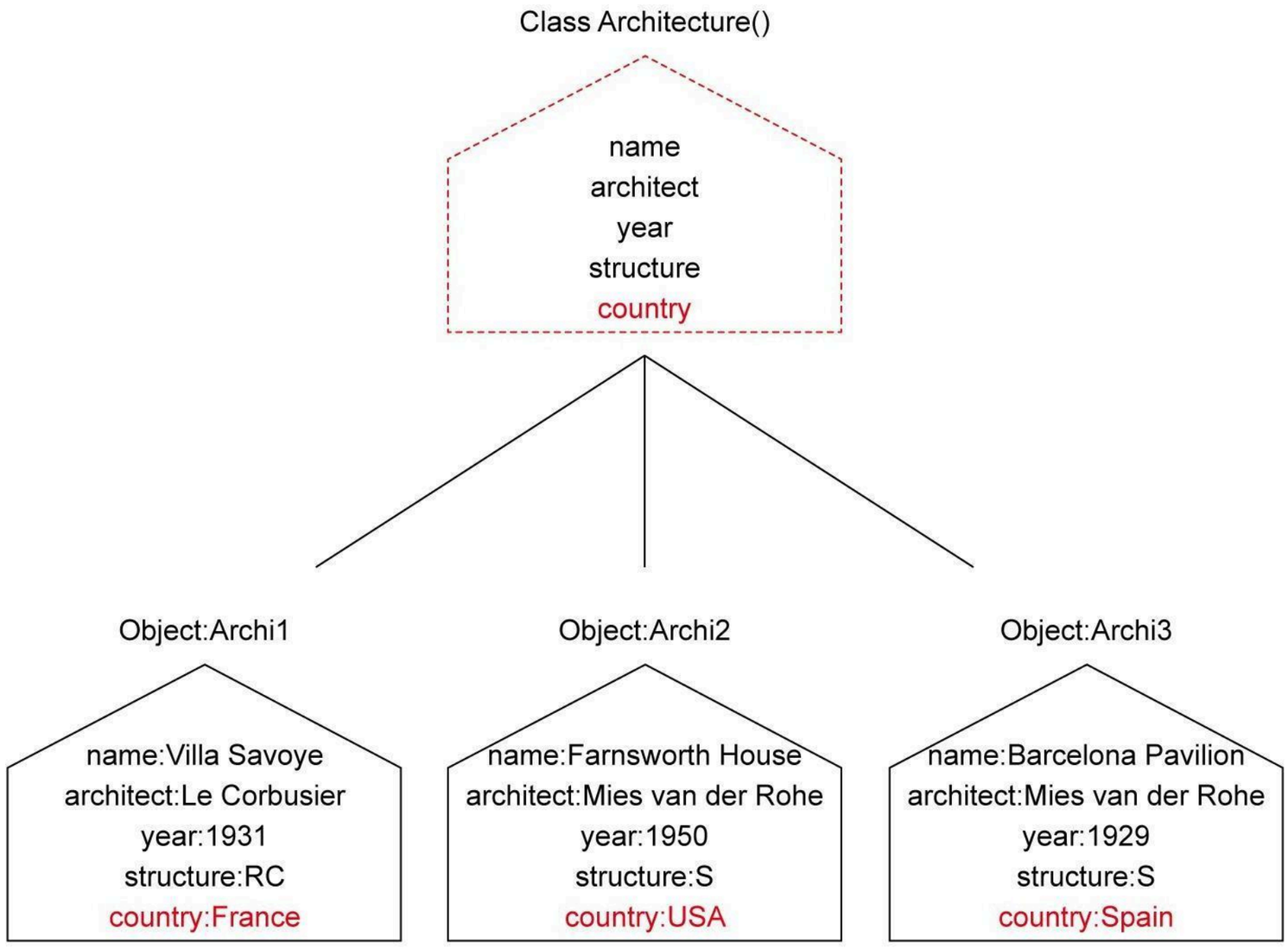
Develop Architecture Class 1

Please define new class function “compareStructure()” in Architecture() Class.

- class function: compareStructure()
    - Argument value: Architecture Class object
    - Console output: Display whether two Architecture are the same structure or not.

Develop Architecture Class 2

Please define new class value “self.country” and class function “checkContinent()” in Architecture() Class.



- class value: self.country:

        - country’s name
- class function: checkContinent()
    - Argument value: None
    - Console output: Determine continent from self.country, and display result

RHINO PYTHON LECTURE | DAY5

1. Contents
    1. Installing TensorFlow
    2. What’s difference between machine leaning and deep learning
    3. MNist

---

1 Installing TensorFlow

1.1 Install Python into windows

If you can work python on rhino and grasshopper, you have to install.

Download install package from following url and execute the file.

https://www.python.org/ftp/python/3.6.2/python-3.6.2-amd64.exe

Open Comannd Prompt.

If you enter ‘Python’ on the prompt, python console start.

Command Prompt - python

```
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:¥Users¥Masamiki>python
Python 3.6.2 (v3.6.2:5fd33b5, Jul  8 2017, 04:57:36) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

attention:

If installation is success but you can't up python console, you have to add python's path to windows environmental variable.

python's path is following.

'C:\Users*{your username}*\AppData\Local\Programs\Python\Python36′

reference: *https://stackoverflow.com/questions/3701646/how-to-add-to-the-pythonpath-in-windows*

1.2 Install tensorflow

On the command prompt enter following command.

```
pip3 install --upgrade tensorflow
```

Command Prompt - pip3 install --upgrade tensorflow

```
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:¥Users¥Masamiki>python
Python 3.6.2 (v3.6.2:5fd33b5, Jul  8 2017, 04:57:36) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 1 + 1
2
>>> import tensorflow as ts
>>> exit
Use exit() or Ctrl-Z plus Return to exit
>>> exit
Use exit() or Ctrl-Z plus Return to exit
>>> exit()

C:¥Users¥Masamiki>pip3 install --upgrade tensorflow
Collecting tensorflow
  Downloading https://files.pythonhosted.org/packages/e7/88/417f18ca7eed5ba9bebd51650d04a4af9
/tensorflow-1.9.0-cp36-cp36m-win_amd64.whl (37.1MB)
    100% |████████████████████████████████| 37.1MB 996kB/s
```

if installation is done without any trouble,
you can use following command on the python console.

?

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
```

attention:

if you can’t import tensorflow, there is also the possiblirity that script path is not included in windows environmental variable.

The script path is following.

‘c:\users*{your username}*\appdata\local\programs\python\python36\Scripts’

reference: *https://stackoverflow.com/questions/3701646/how-to-add-to-the-pythonpath-in-windows*

---

2. GH_CPython

2.1 Path of Python 3.6

Open Command Prompt(if you have a terminal application, you can open whichever you want to use).

‘where’ command finds path with the argument that you provide.

Check the path of python3.6 and remember it to use python3.6 in grasshopper.

cmd

<1> cmd

Search

```
$
Masamiki@DESKTOP-0TV222U C:\Users\Masamiki
$
Masamiki@DESKTOP-0TV222U C:\Users\Masamiki
$
Masamiki@DESKTOP-0TV222U C:\Users\Masamiki
$
Masamiki@DESKTOP-0TV222U C:\Users\Masamiki
$
Masamiki@DESKTOP-0TV222U C:\Users\Masamiki
$
Masamiki@DESKTOP-0TV222U C:\Users\Masamiki
$
Masamiki@DESKTOP-0TV222U C:\Users\Masamiki
$
Masamiki@DESKTOP-0TV222U C:\Users\Masamiki
$
Masamiki@DESKTOP-0TV222U C:\Users\Masamiki
$
Masamiki@DESKTOP-0TV222U C:\Users\Masamiki
$ where python
C:\Users\Masamiki\AppData\Local\Programs\Python\Python36\python.exe
C:\Python27\python.exe

Masamiki@DESKTOP-0TV222U C:\Users\Masamiki
$
```

cmd.exe*[64]:2508 « 180626[64] 1/1 [+] NUM PRI↕ 80x26 (3,143) 25V

2.2 Install GH_Cpython

We can use limited libraries in GH_Python. We can't use other libraries that we want to use except libraries prepared by Rhinoceros.

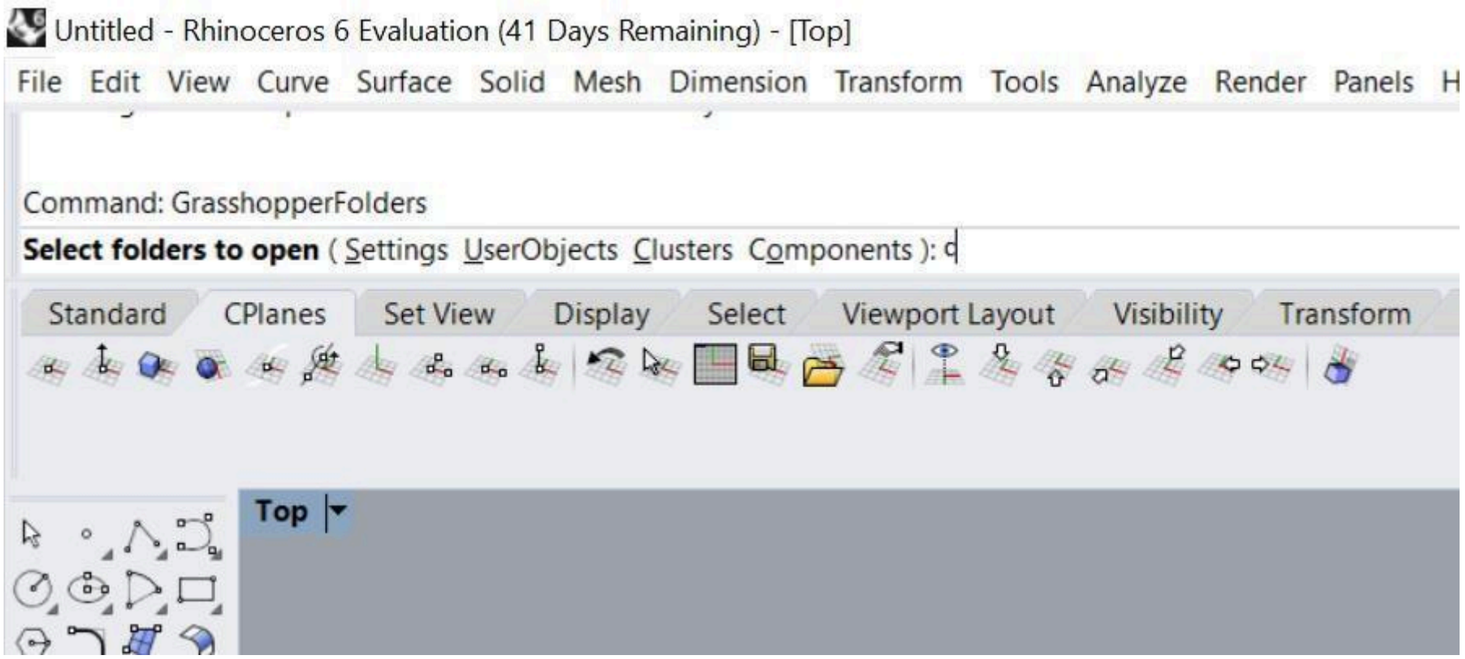
If we want to import other libraries, we have to use GH_CPython.

here, we can download GH_CPython

https://www.food4rhino.com/app/ghcpython

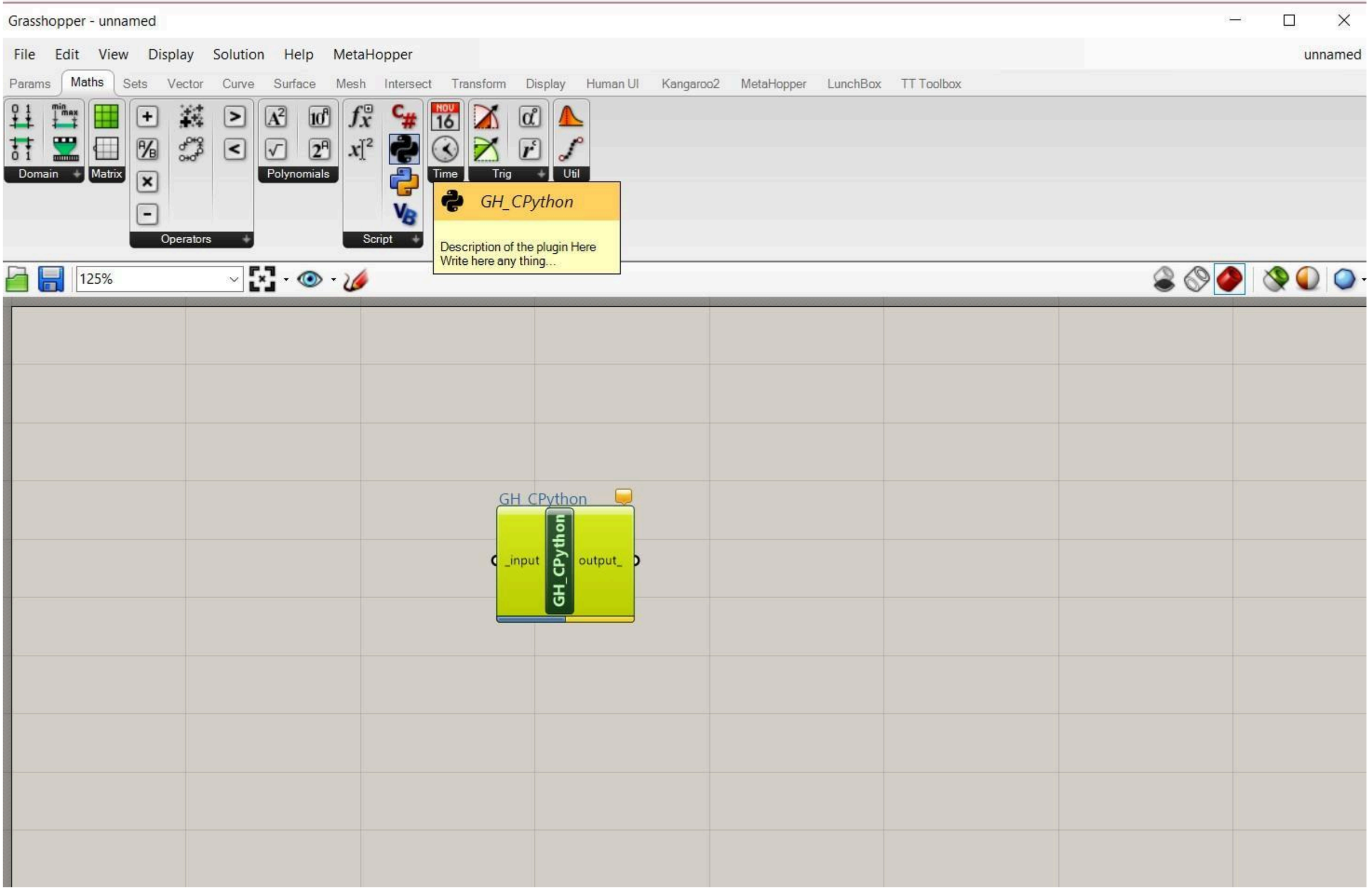
If you forget the folder in where you have to put plugin files, enter 'GrasshopperFolders' and then 'c' on the Rhino.

you can see the folder for plugin.



2.3 Setting

After launching Grasshopper, you can chose GH_CPython.


Grasshopper - unnamed
File
Edit
View
Display
Solution
Help
MetaHopper
unnamed
Params
Maths
Sets
Vector
Curve
Surface
Mesh
Intersect
Transform
Display
Human UI
Kangaroo2
MetaHopper
LunchBox
TT Toolbox
Domain
Matrix
Operators
Polynomials
Script
Time
Trig
Util
GH_CPython
Description of the plugin Here
Write here any thing...
125%
GH_CPython
_input
GH_CPython
output_

Select ‘Choose Interpreter’ on the menu after opening the editor by double click.

GH_CPython

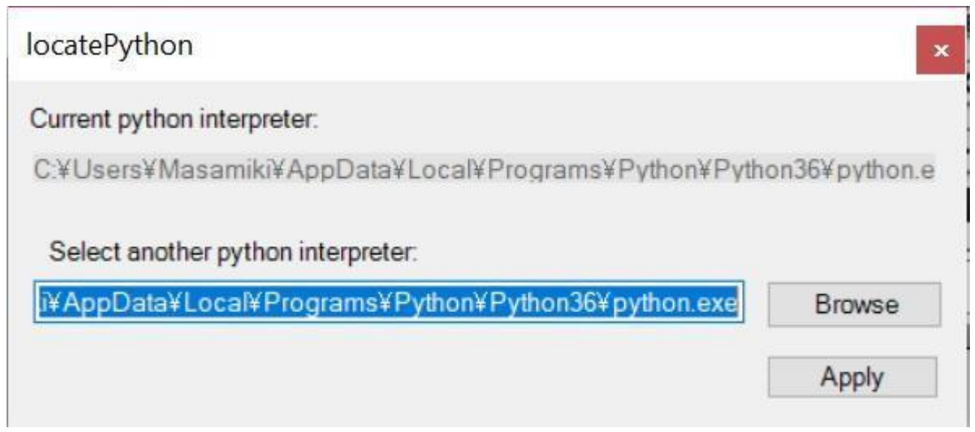
File Data Python

Choose Interpreter
Pip install
Python shell

```
# -*-
"""
Pythor
Created on  Monday July 2018 01:11:42
@author:  Masamiki

[desc]
Description of the plugin Here
Write here any thing...
[/desc]

ARGUMENTS:
----------
<inp>
    _input :[required] - [type = int] - [default = None]
    Descripe your input here
        * bullet point.
        * bullet point
</inp>
<inp>
    Other inputs go here ...
</inp>

RETURN:
----------
    <out>
```

```
Traceback (most recent call last):
  File "C:\GH_CPython\PythonFileWritten_1.py", line 33, in <module>
    import numpy
ImportError: No module named numpy
```

Please input your Python3.6 path into the input field.



Now you can use python3.6 and TensorFlow on the GH_CPython.

Try this following code.

```
try:
 import tensorflow as tf
 hello = tf.constant(_input)
 sess = tf.Session()
 print(sess.run(hello))
 output_ = 'success'
except:
 output_ = 'failed'
```

RHINO PYTHON LECTURE | DAY5: TENSORFLOW

Difference of Machine Learning and Deep Learning

Machine Learning
Data -> feature(HOG, SIFT) -> learning(SVN, KNN) -> result

Hog and SIFT are feature which is extracted from brightness slope distribution of a cell. In the machine learning process, we need the feature created by human for learning.

Deep learning
Data -> neural network(deep leaning) -> result

Deep leaning is also called ‘end to end machine learning’. In this process we don’t need to prepare the feature data. it’s created by neural network.

Key words

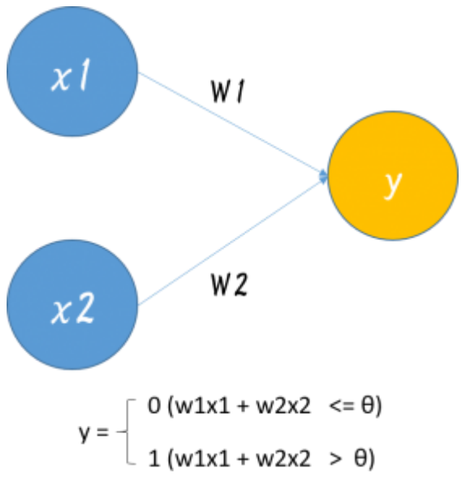
Training data / Test data

Training data and Test data have data which we want analyze, and answer data.

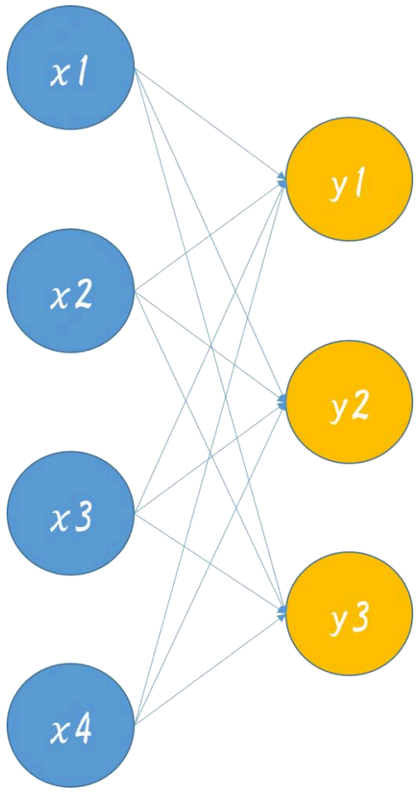
Training data is the data that create feature and learned data(formula?, AI?)

Test data is the data for check the learned data. We give test data to learned data and compare result and answer data.

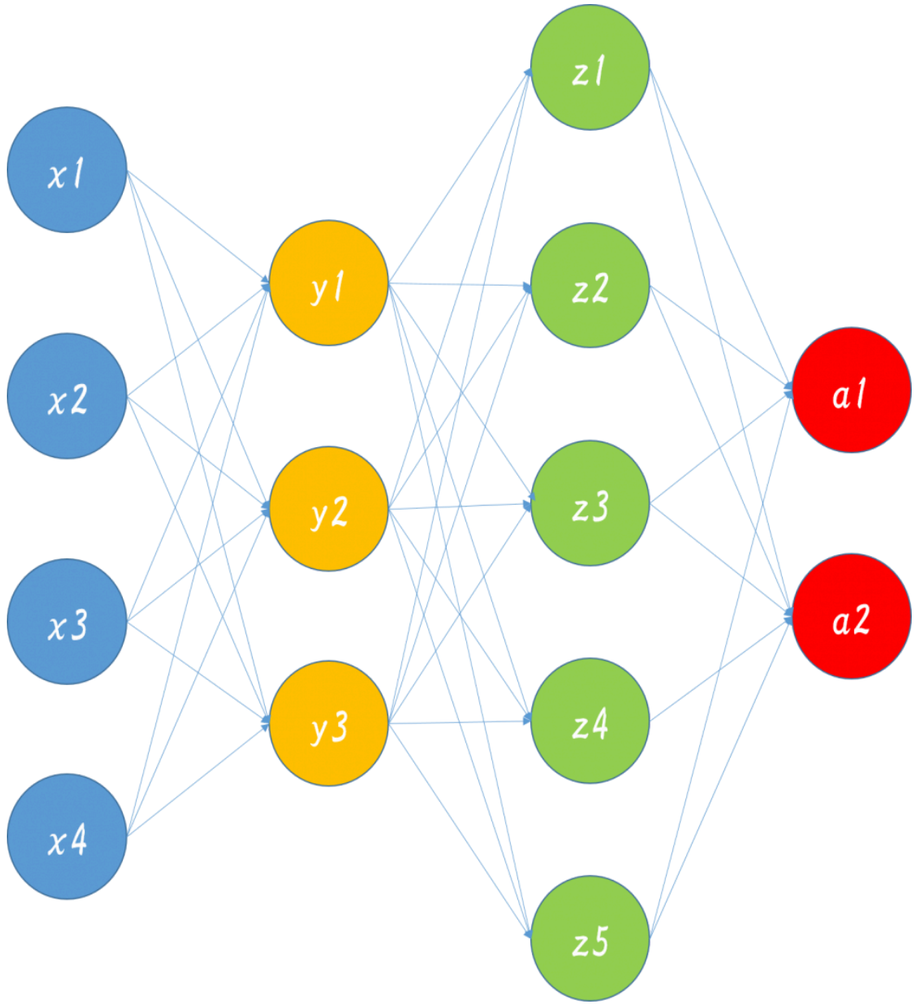
Perceptron



‘Perceptron’ is a mechanism that get value and output value. In perceptron, input values like x1 and x2 are multiplied by weight like w1 and w2. Sum of those values is evaluated to determine y’s value.



The bunch of Perceptrons is called layer.



Deep learning work on bunch of layers (deep layers).

※Actually y is not determined by x and weight but also bias. this article is simplified for beginner.

Softmax

$$p_i = \frac{e^{y_i}}{\sum_{k=1}^{n} e^{y_k}}$$

softmax is like a formula which create rate.

```
>>> import math
>>> z = [1.0, 2.0, 3.0, 4.0, 1.0, 2.0, 3.0]
>>> z_exp = [math.exp(i) for i in z]
>>> print([round(i, 2) for i in z_exp])
[2.72, 7.39, 20.09, 54.6, 2.72, 7.39, 20.09]

>>> sum_z_exp = sum(z_exp)
>>> print(round(sum_z_exp, 2))
114.98

>>> softmax = [i / sum_z_exp for i in z_exp]
>>> print([round(i, 3) for i in softmax])
[0.024, 0.064, 0.175, 0.475, 0.024, 0.064, 0.175]
```

this is useed to derive the category of the data.

If you analyse image with deep learning in which used softmax, the result is like following

Dog: 2.4%

Cat: 6.4%

Monkey: 17.5%

….

Tensor

Tensor is a similar concept to vector but it include vector.

A vector is represented as a 1-dimensional array in a basis, and is a 1st-order tensor. Scalars are single numbers and are thus 0th-order tensors.

Practice: MNIST

MNIST is a basic technique of machine learning.

Download following files.

train-images-idx3-ubyte.gz: `training set images (9912422 bytes)`
train-labels-idx1-ubyte.gz: `training set labels (28881 bytes)`
t10k-images-idx3-ubyte.gz: `test set images (1648877 bytes)`
t10k-labels-idx1-ubyte.gz: `test set labels (4542 bytes)`

images: a lot of 28 x 28 images drawn numbers by hand.

label: text data that indicate what number is drawn in the images like 3 or 4.

?

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data mnist
= input_data.read_data_sets("MNIST_data/", one_hot=True)

```

It might raise warning, but you don’t have to care.

‘mnist’ variable has processed data by read_data_sets method for mnist.

?

```
# check the number of pair for traninng.
print(mnist.train.num_examples)
# check the label data
print(mnist.train.labels)

```

```
</pre>
# check the number of pair for traninng.
print("------show number       ")
print(mnist.train.num_examples)
print("                         ")

# check the label data
print("------labes       ")
print(mnist.train.labels)
print("                         ")
<pre>
```

let’s create neural network.

?

```
# input value
# 'placeholder' is a place which can have one image. (It's converted into array which has 28 * 28 = 784 values.) Here 'None'means that a dimension can be of any length. x =
tf.placeholder(tf.float32, [None, 784])

# weight
# [784, 10] because we want to multiply the 784-dimensional image vectors by it to produce 10-dimensional vectors of evidence for the difference classes W
= tf.Variable(tf.zeros([784, 10]))

# bias
b = tf.Variable(tf.zeros([10]))

# We can now implement our model. It only takes one line to define it! #
x by W with the expression tf.matmul(x, W)
y = tf.matmul(x, W) + b
```

```
y_ = tf.placeholder(tf.float32, [None, 10])
cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y))

```

let’s start learning

?

```
# compare the result of calculation with image data, which has 1 * 10 array, and the data which is in labes.
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
sess = tf.InteractiveSession() tf.global_variables_initializer().run()

# Training: give the 100 data choosen from 55000 data into neural network 1000 times for
_ in range(1000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})

```

let’s evaluate the reuslt of trainning with test data.

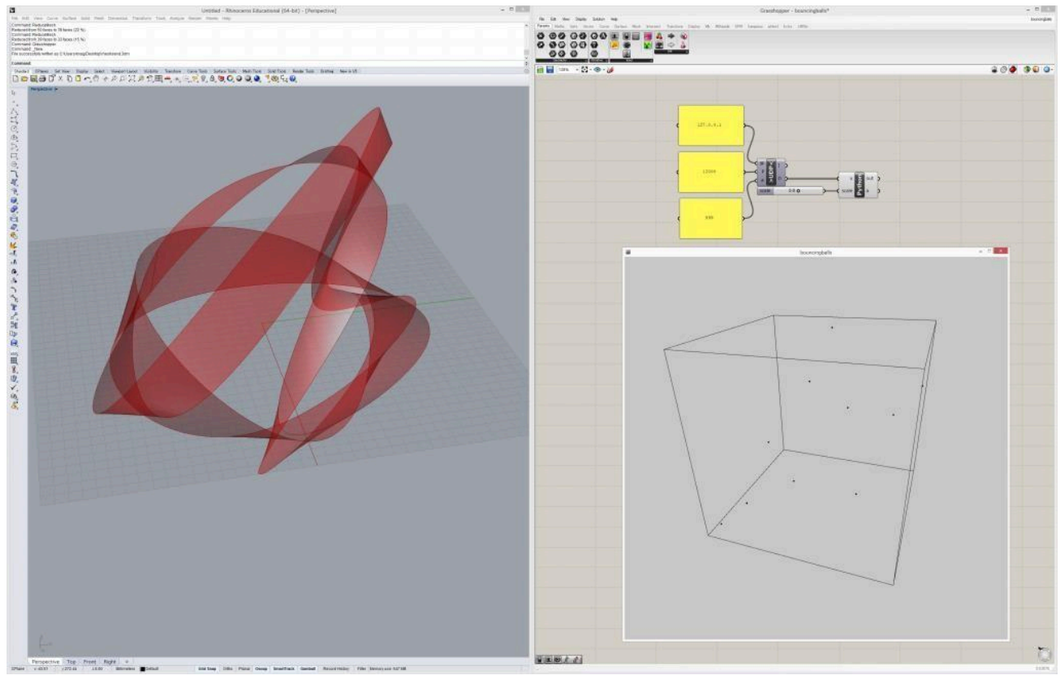
```
# evaluate the reuslt of trainning with test data. correct_prediction
= tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy_var = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

accuracy = sess.run(accuracy_var, feed_dict={x: mnist.test.images, y_: mnist.test.labels})
```

PROCESSING+GHPYTHON PRACTICE #00 | COMMUNICATION

1. Practice | *answers*
    1. Task1 – processing -> python
        1. Communication
        2. Pipeline | Processing
        3. Pipeline | Python
        4. Hints | Processing
        5. Hints | Python
    2. Task2 – python -> processing
        1. Communication
        2. Pipeline | Processing
        3. Pipeline | Python
        4. Hints | Processing
        5. Hints | Python

Practice

Task1 – processing -> python



Communication

- Processing -> Python

        - All the positions of particles within
        - All the velocities of particles(Vector)

Pipeline | Processing

- Declaration
    - OscP5, NetAdress
    - PeasyCam
    - Amount of particles, speed
    - List of PVectors for position and velocity
- Init
    - Init OscP5, NetAdress
    - Init PeasyCam
    - Init List of PVectors
- Function | sendOSC
    - define OscMessage with title “/osc”
    - make a list for strings for positions
    - define strings for positions and substitute it to the list
    - join strings and add it to the OscMessage
    - make a list for strings for velocities
    - define strings for positions and substitute it to the list
    - join strings and add it to the OscMessage
    - send OscMessage
- loop
    - fill background with white
    - draw every point on canvas
    - if points are out side of canvas, inverse velocities
    - sendOSC

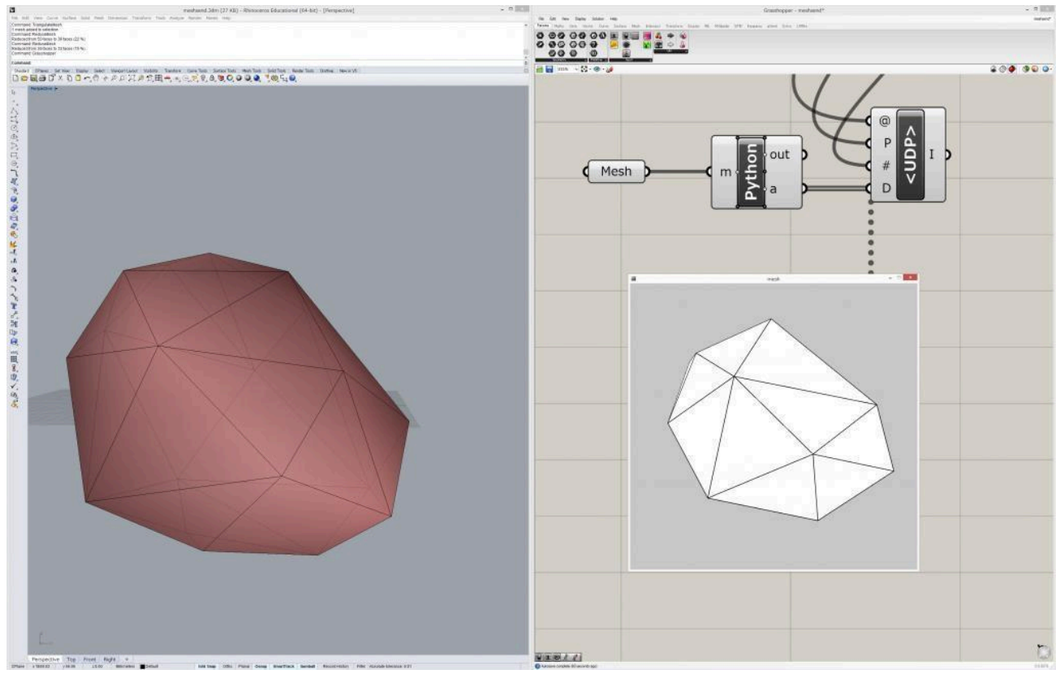
Pipeline | Python

- import
    - import rhinoscriptsyntax

- function | convertStringToPoints
  - declare list of points
  - split string into strings
  - for each strings split strings and generate points
  - append points to the list
  - return points
- generation
  - declare list for lines
  - generate position point from string
  - generate velocity point from string
  - for each points in each list draw line between p and p+v

- append lines to the list
- generate loft surface with lines

Task2 – processing -> python



Communication

- Python -> Processing
  - Verts’ infomation
  - Faces’ information

Pipeline | Python

- import
  - import rhinoscriptsyntax
- generation
  - generate mesh verts
  - generate mesh faces
  - declare list <messages>
  - append <“/osctest”>to <messages>
  - declare empty text variable <message>

- declare list <ss>
- for every verts, generate text and append it in <ss>
- join <ss> with “/” and add it to <message>
- add “:” to <message>

- initialize <ss>
- for every faces, generate text and append it in <ss>
- join <ss> with “/” and add it to <message>
- append <message> in <messages>

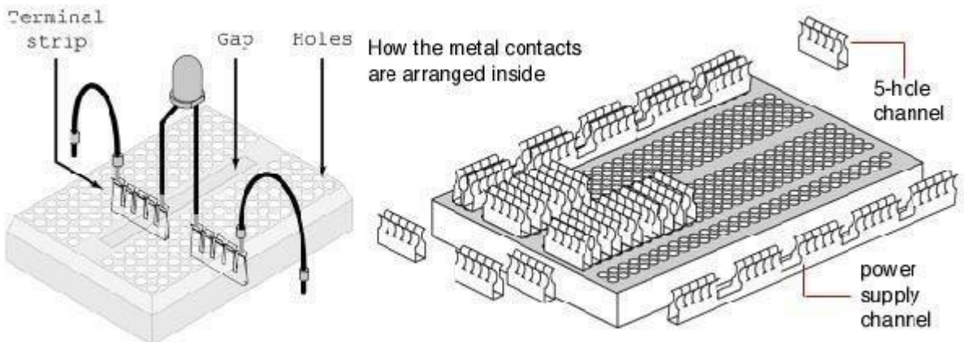
Pipeline | Processing

ARDUINO TUTORIAL #01 | INPUT / OUTPUT

Contents

Getting Started

Bread board



images from http://cuartielles.com/verkstad/edu/

Bread board enable us to build circuits without soldering. We can connect pins by insert them into holes of bread boards.

Output

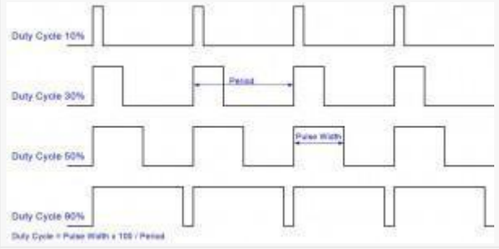
There are two functions for output:

- digitalWrite(,)
- analogWrite(,)

Digital out – Blink

```
int ledPin = 13;
int interval = 1000;
void setup() {
 pinMode(ledPin, OUTPUT);
}

void loop() {
 digitalWrite(ledPin, HIGH);
 delay(interval);
 digitalWrite(ledPin, LOW);
 delay(interval);
}
```

Analog out - fade



atmega168a_pwm_02_lrg

```
In fact, Arduino doesn't have the function to output analog value. To control LED (or other output devices t
example DC motors) gradually , we can use "PWM - Pulse Width Modulation" as a control method. By PWM contro
can change apparent voltage by changing duty rate instead of changing peek voltage. For example, when the p
voltage is fixed in 5V, 10% duty rate appears as 0.5V.
```

We can use analogWrite function for PWM control.

```
int ledPin = 9;
int step = 5;
int val;
void setup() {
 pinMode(ledPin, OUTPUT);
 val = 0;
}

void loop() {
 analogWrite(ledPin, val);
 val+=step;
 if(val>255-step || val<step){
  step*=-1;
 }
 delay(10);
}
```

Analog out - servo

We can control servos by using PWM as well.

```
int servoPin = 9;
int step = 32;
int val;
void setup() {
 pinMode(servoPin, OUTPUT);
 val = 0;
}
```

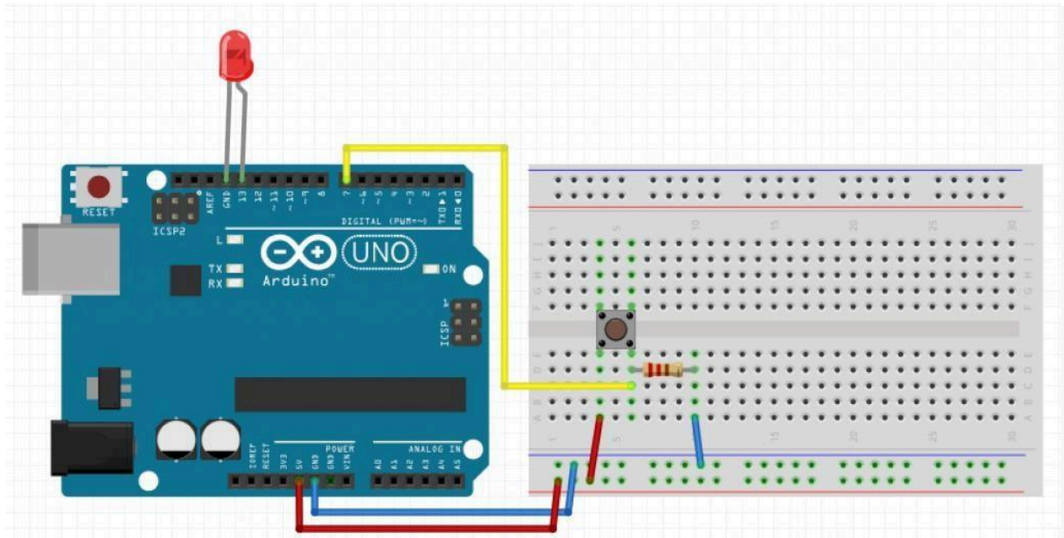
```
7
8       void loop() {
9        analogWrite(servoPin, val-1);
10       val+=step;
11       if(val>256-step || val<step){
12        step*=-1;
13       }
14       delay(500);
15      }
16
```

Input

Same as output, there are two functions for input:

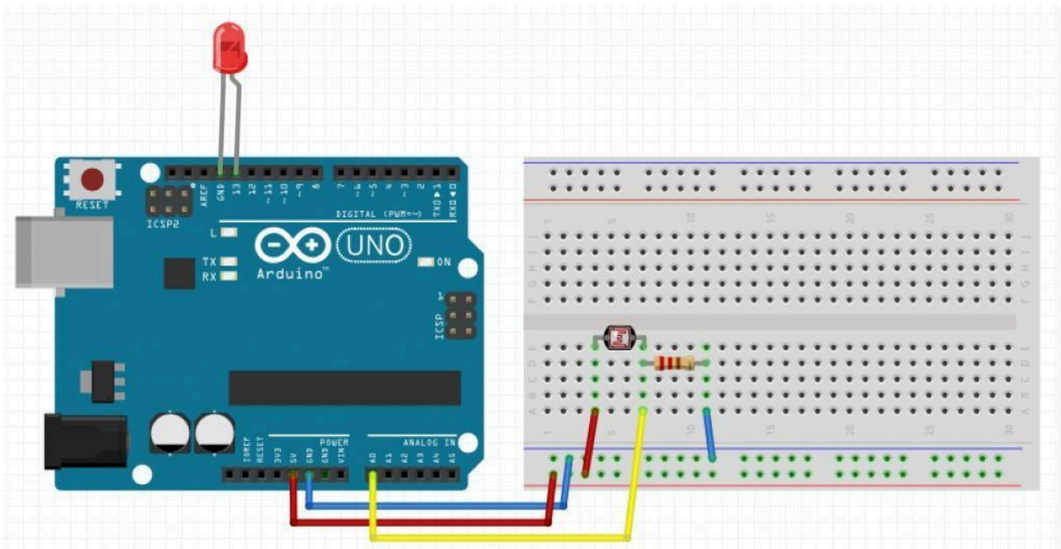
- digitalRead(<pin number>)
- analogRead(<pin number>)

Digital read - Tact switch




```
int ledPin = 13;
int swPin = 7;
void setup() {
 pinMode(ledPin, OUTPUT);
 pinMode(swPin, INPUT);
}

void loop() {
 digitalWrite(ledPin, digitalRead(swPin));
}
```

Analog read - CdS cell (Light sensor)




```
int ledPin = 11;
int cdsPin = A0;
void setup() {
 pinMode(ledPin, OUTPUT);
 pinMode(cdsPin, INPUT);
}

void loop() {
 analogWrite(ledPin, analogRead(cdsPin)/4);
}
```

```
int ledPin = 11;
int cdsPin = A0;
void setup() {
 Serial.begin(9600);
 pinMode(ledPin, OUTPUT);

 pinMode(cdsPin, INPUT);
```

```
}

void loop() {
 Serial.println(analogRead(cdsPin));
 int value = map(analogRead(cdsPin), 0, 300, 0, 50);
 analogWrite(ledPin, value);
}
```

ARDUINO TUTORIAL #02 | SERIAL COMMUNICATION

Contents

Basic

Functions

- Serial.begin(<band>)
- Serial.available()
- Serial.read()
- Serial.print(<message>)
- Serial.println(<message>)

Hello, Arduino

[?]

```
void setup(){
 Serial.begin(9600);
}


void loop(){
}

void serialEvent() {
```

```
 if(Serial.available()>0){
  if(Serial.read()=='h'){
   Serial.println("Hello, I am Arduino");
  }
 }
}
```

Blink

[?]

```
int led = 13;

void setup(){
 Serial.begin(9600);
 pinMode(led, OUTPUT);
}

void loop(){
}

void serialEvent() {
 if(Serial.available()>0){
  char c = Serial.read();
  if(c=='h'){
   digitalWrite(led,HIGH);
   Serial.println("I turned led on!");
  }
  if(c=='l'){
   digitalWrite(led,LOW);
   Serial.println("I turned led off!");
  }
 }
```

```
}
```

Use
Long message
?

```
String inputString = "";
boolean stringComplete = false;

void setup() {
 Serial.begin(9600);
}

void loop() {
 if (stringComplete) {
 if(inputString == "hello"){
 Serial.println("Hello, I'm Arduino.");
 }else if(inputString == "bye"){
 Serial.println("See you.");
 }else{
 Serial.println("?");
 }
 inputString = "";
 stringComplete = false;
 }
}


void serialEvent() {
 while (Serial.available()) {
 char inChar = (char)Serial.read();
 if (inChar == '\n') {
 stringComplete = true;
```

```
 }else{
 inputString += inChar;
 }
 }
}
```

+Processing

Arduino code

?

```
int led = 9;

void setup() {
 Serial.begin(9600);
 pinMode(led,OUTPUT);
}

void loop() {
}

void serialEvent() {
 if(Serial.available()>0){
 analogWrite(led, int(Serial.read()));
 }
}
```

processing code

?

```
import processing.serial.*;

Serial myPort;

void setup() {
 size(255, 255);
 println(Serial.list());
 strokeWeight(5);
 String portName = Serial.list()[2]; myPort
 = new Serial(this, portName, 9600);
}

void draw() {
 background(255);
 myPort.write(int(mouseX));
 point(mouseX,mouseY)
}
```

Contents

Libraries

There are several libraries specified to the each modules for Grove. These libraries are available at github repository of reedstudio. Please access to ***the repository*** and download them. This package of libraries contain other libraries for sensors and actuators as well.

Accelerometer

The library for the accelerometer by Grove is ” DigitalAccelerometer_MMA7660FC “. You can add it to Arduino by renaming it ” MMA7660FC ” and placing it to the library folder in Arduino folder.

?

```
#include <Wire.h>
#include <MMA7660.h>
```

```
MMA7660 accelemeter;

void setup(){
 accelemeter.init();
 Serial.begin(9600);
}

void loop(){
 float
 ax,ay,az;

 accelemeter.getAcclemeter(&ax,&ay,&az);

 Serial.print("x = ");
 Serial.println(ax);
 Serial.print("y = ");
 Serial.println(ay);
 Serial.print("z = ");
 Serial.println(az);

 Serial.println("--");
 delay(200);
}
```

Ultra sonic ranger finder

The library for the Ultra sonic range finder by Grove is ” Ultrasonic “. You can add it to Arduino by placing it to the library folder in Arduino folder.

?

```
#include <NewSoftSerial.h>
#include <Ultrasonic.h>

Ultrasonic ultrasonic(7);
```

```
5
6   void setup(){
7     Serial.begin(9600);
8   }
9
10  void loop(){
11    ultrasonic.MeasureInCentimeters();
12    delay(150);
13    Serial.print("The distance is ");
14    Serial.print(ultrasonic.RangeInCentimeters);
15    Serial.println(" cm");
16  }
```

ARDUINO TUTORIAL #02 | SERIAL COMMUNICATION

Contents

Basic

Functions

- Serial.begin(<band>)
- Serial.available()
- Serial.read()
- Serial.print(<message>)
- Serial.println(<message>)

Hello, Arduino

?

```
1   void setup(){
2    Serial.begin(9600);
3   }
```

```
void loop(){
}

void serialEvent() {
 if(Serial.available()>0){
  if(Serial.read()=='h'){
   Serial.println("Hello, I am Arduino");
  }
 }
}
```

Blink

[?]

```
int led = 13;

void setup(){
 Serial.begin(9600);
 pinMode(led, OUTPUT);
}

void loop(){
}

void serialEvent() {
 if(Serial.available()>0){
  char c = Serial.read();
  if(c=='h'){
   digitalWrite(led,HIGH);
   Serial.println("I turned led on!");
  }
  if(c=='l'){
```

```
    digitalWrite(led,LOW);
    Serial.println("I turned led off!");
   }
  }
}
```

Use

Long message

?

```
String inputString = "";
boolean stringComplete = false;

void setup() {
 Serial.begin(9600);
}

void loop() {
 if (stringComplete) {
 if(inputString == "hello"){
 Serial.println("Hello, I'm Arduino.");
 }else if(inputString == "bye"){
 Serial.println("See you.");
 }else{
 Serial.println("?");
 }
 inputString = "";
 stringComplete = false;
 }
}

void serialEvent() {
```

```
  while (Serial.available()) {
  char inChar = (char)Serial.read();
  if (inChar == '\n') {
  stringComplete = true;
  }else{
  inputString += inChar;
  }
  }
 }
```

+Processing

Arduino code

[2]

```
int led = 9;


void setup() {
 Serial.begin(9600);
 pinMode(led,OUTPUT);
}

void loop() {
}

void serialEvent() {
 if(Serial.available()>0){
 analogWrite(led, int(Serial.read()));
 }
}
```

processing code

?

```
import processing.serial.*;

Serial myPort;

void setup() {
 size(255, 255);
 println(Serial.list());
 strokeWeight(5);
 String portName = Serial.list()[2]; myPort
 = new Serial(this, portName, 9600);
}

void draw() {
 background(255);
 myPort.write(int(mouseX));
 point(mouseX,mouseY)
}
```

ARDUINO TUTORIAL #03 | USING SENSORS WITH LIBRARIES

Contents

Libraries

There are several libraries specified to the each modules for Grove. These libraries are available at github repository of reedstudio. Please access to ***the repository*** and download them. This package of libraries contain other libraries for sensors and actuators as well.

Accelerometer

The library for the accelerometer by Grove is ” DigitalAccelerometer_MMA7660FC “. You can add it to Arduino by renaming it ” MMA7660FC ” and placing it to the library folder in Arduino folder.

?

```
#include <Wire.h>
#include <MMA7660.h>
MMA7660 accelemeter;

void setup(){
 accelemeter.init();
 Serial.begin(9600);
}

void loop(){
 float
 ax,ay,az;

 accelemeter.getAcclemeter(&ax,&ay,&az);

  Serial.print("x = ");
  Serial.println(ax);
  Serial.print("y = ");
  Serial.println(ay);
  Serial.print("z = ");
  Serial.println(az);

  Serial.println("--");
  delay(200);
 }
```

Ultra sonic ranger finder

The library for the Ultra sonic range finder by Grove is ” Ultrasonic “. You can add it to Arduino by placing it to the library folder in Arduino folder.

?

```
#include <NewSoftSerial.h>
#include <Ultrasonic.h>

Ultrasonic ultrasonic(7);

void setup(){
  Serial.begin(9600);
}

void loop(){
  ultrasonic.MeasureInCentimeters();
  delay(150);
  Serial.print("The distance is ");
  Serial.print(ultrasonic.RangeInCentimeters);
  Serial.println(" cm");
}
```

PROCESSING ADVANCED INTRO #01 | OBJECT-ORIENTED PROGRAMING

Contents

What’s OOP?

Outline

Object-Oriented Programing (OOP) is one of the method for programing. The one we used in previous tutorial is Procedural Programing (PP). OOP is the method to define “object” which has several “properties” and “behaviors” in contrast to PP is the method to make workflow with variables and functions.

These two codes are examples showing difference between OOP and PP.

PP

?

```
int x,y,r;

void setup(){
  size(200,200);
  x = 100;
  y = 80;
  r = 20;
}

void draw(){
  background(0);
  ellipse(x,y,r,r);
}
```

OOP

?

```
class
  Circle{
  int x,y,r;

  Circle(int _x, int _y, int
    _r){ x = _x;
    y = _y;
    r = _r;
  }

  void show(){
    ellipse(x,y,r,r);
  }
```

```
}

Circle c;

void setup(){
  size(200,200);
  c = new Circle(100,80,20);
}

void draw(){
  background(0);
  c.show();
}
```

Class

Class is used for defining “objects”. Class is consisted of 3 elements – property variable, method, constructor. Property variables are used for defining properties of the object, method for behavior, and constructor is for initialization.

?

```
class
  Circle{
  int x,y,r;

  Circle(int _x, int _y, int
    _r){ x = _x;
    y = _y;
    r = _r;
  }

  void show(){
    ellipse(x,y,r,r);
  }
```

```
12
13      }
14
```

Practice

RandomWalker

Thinking about objects which walks randomly, we can figure out that it needs following properties and behaviors.

- Property
  - position
  - Step size
- Behavior
  - walk
  - show

?

```
1       //difining object
2       class RandomWalker
3       {
4         //property variables
5         PVector pos;
6         int step;
7
8         //constructor
9         RandomWalker(PVector _pos, int _step) {
10          pos = _pos;
11          step = _step;
12        }
13
14        //methods
15        void
16        walk(){
17          PVector vec = PVector.random2D();
            vec.mult(step);
            pos.add(vec);
          }
          void show() {
```

```
    strokeWeight(5);
    point(pos.x, pos.y);
  }
}

RandomWalker w;

void setup() {
  size(200, 200);
  //initializing object
  w = new RandomWalker(new PVector(100,100), 5);
}

void draw() {
  background(255);
  //call object's behavior
  w.walk();
  w.show();
}
```

PYTHON ADVANCED INTRO #01 | OBJECT-ORIENTED PROGRAMING

Contents

What’s OOP?

Outline

Object-Oriented Programing (OOP) is one of the method for programing. The one we used in previous tutorial is Procedural Programing (PP). OOP is the method to define “object” which has several “properties” and “behaviors” in contrast to PP is the method to make workflow with variables and functions.

These two codes are examples showing difference between OOP and PP.

PP

?

```
import rhinoscriptsyntax as rs

verts = []

verts.append(rs.AddPoint(pt.X-width/2,pt.Y-height/2,0))
verts.append(rs.AddPoint(pt.X+width/2,pt.Y-height/2,0))
verts.append(rs.AddPoint(pt.X+width/2,pt.Y+height/2,0))
verts.append(rs.AddPoint(pt.X-width/2,pt.Y+height/2,0))
verts.append(rs.AddPoint(pt.X-width/2,pt.Y-height/2,0))

a = rs.AddPolyline(verts)
```

OOP

?

```
import rhinoscriptsyntax as rs
class Rect:
    def__init__(self,  pt,  width,  height):
        self.center = pt
        self.w  =  width
        self.h        =
        height
    def
        geometry(self)
        : verts = []
        verts.append(rs.AddPoint(self.center.X-self.w/2,self.center.Y-self.h/2,0))
        verts.append(rs.AddPoint(self.center.X+self.w/2,self.center.Y-self.h/2,0))
        verts.append(rs.AddPoint(self.center.X+self.w/2,self.center.Y+self.h/2,0))
        verts.append(rs.AddPoint(self.center.X-self.w/2,self.center.Y+self.h/2,0))
        verts.append(rs.AddPoint(self.center.X-self.w/2,self.center.Y-self.h/2,0))
        return rs.AddPolyline(verts)

a = Rect(pt,w,h)
```

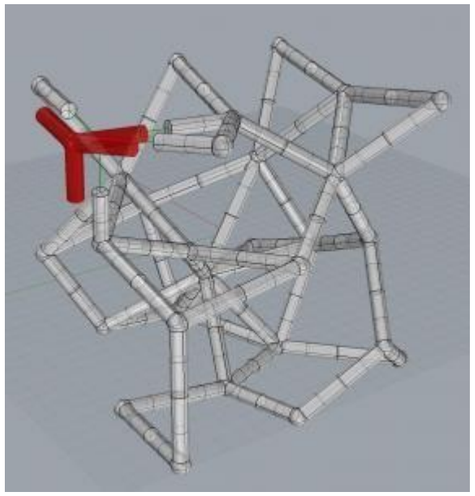
Class

Class is used for defining “objects”. Class is consisted of 3 elements – property variable, method, constructor. Property variables are used for defining properties of the object, method for behavior, and constructor is for initialization.

```
class Rect:
    //constructor
    def_init_(self, pt, width, height):
        //property
        variables
        self.center = pt
        self.w = width
        self.h = height
    //methods
    def
        geometry(self)
        : verts = []
        verts.append(rs.AddPoint(self.center.X-self.w/2,self.center.Y-self.h/2,0))
        verts.append(rs.AddPoint(self.center.X+self.w/2,self.center.Y-self.h/2,0))
        verts.append(rs.AddPoint(self.center.X+self.w/2,self.center.Y+self.h/2,0))
        verts.append(rs.AddPoint(self.center.X-self.w/2,self.center.Y+self.h/2,0))
        verts.append(rs.AddPoint(self.center.X-self.w/2,self.center.Y-self.h/2,0))
        return rs.AddPolyline(verts)
```

Practice

Component system



Thinking about component based structures, it has several geometric properties and shapes.

The Component which is shown at the right, has center point and vectors to the connected point as geometric properties and skeleton lines and pipe based shape as shape.

- Property
    - center point
    - target vectors
- Behavior

- generate skeleton
- generate pipe geometries

```
import ghpythonlib.components as gh
class component:
    def_init_(self, center, targets, range): self.center
        = center
        self.vectors = []
        for pt in
        targets:
            dist = gh.Distance(center,pt)
            if dist!=0 and dist<range:
                vec = gh.Vector2Pt(center,pt)
                self.vectors.append(vec)

    def
        skeleton(self)
        : lines = []
        for vec in self.vectors:
            length = gh.VectorLength(vec)
            line = gh.LineSDL(self.center, vec, length*0.5)
            lines.append(line)
        return lines

    def geometry(self,radius):
        geos = []
        sph = gh.Sphere(gh.XYPlane(self.center),radius*1.2)
        geos.append(sph)
        for vec in self.vectors:
            length = gh.VectorLength(vec)
            line = gh.LineSDL(self.center, vec, length*0.5)
            pipe = gh.Pipe(line,radius,1) geos.append(pipe)
        geo = gh.SolidUnion(geos)
        return geo

a = component(compPt, Pts , range)
```

PROCESSING ADVANCED INTRO #02 | OBJECT-ORIENTED PROGRAMMING

Contents

Advantages of OOP

Encapsulation

Encapsulation is a packing of data and function into a single component. This is almost what we learned at previous session. Another feature of encapsulation is information hiding. Because one of the targets of OOP is to define object as concept model, some variables and functions don't need to be handled from outside of the Class. For example, thinking about RandomWalker Class, perhaps we don't need to access the variable of position. Also, encapsulation protects variables from unexpected modifications.

You can hide variables and function from outside of Class by using "private" when defining variables or functions.

?

```
class RandomWalker {
  //once we define these variables we don't need to care about them
  private int x;
  private int y;
  private int
  step;
  RandomWalker(int _x,int _y, int
    _step){ x = _x;
    y = _y;
    step = _step;
  }

  void
    update(){
    move();
    display();
  }

  //these functions are needed only insude the class
  private void move(){
    PVector rand = PVector.random2D();
    rand.mult(step);
    x += rand.x;
    y += rand.y;
  }
```

```
23      private void
24        display(){
25        strokeWeight(5);
26        point(x,y);
27      }
28    }
29
30    RandomWalker r;
31
32    void setup(){
33      size(200,200);
34      r = new RandomWalker(100,100,5);
35    }
36
37    void draw(){
38      background(255);
39      r.update();
40      r.display();
41    }
```

Inheritance

Inheritance is a function to make new Class from existing Class with additional variables and functions. The advantage of inheritance is that we can reuse existing code by others to make our programs without reinventing existing code.

For example, you can make Agent(RandomWalker) Class as extension of PVector Class.

?

```
1     class agent extends PVector {
2       private int step;
3       agent(int _x, int _y, int _step) {
4         //constructor of base Class
          super(_x, _y);
```

```
    step = _step;
  }

  void
    update(){
    move();
    display();
  }

  private void move() {
    PVector rand = PVector.random2D();
    rand.mult(step);
    //function of base Class
    super.add(rand);
  }

  private void display()
      { strokeWeight(5);
      point(super.x,super.y);//variables of base Class
    }
  }

  agent a;

  void setup(){
    size(200,200);
    a = new agent(100,100,5);
  }

  void draw(){
    background(255);
    a.update();
  }
```

Practice

Cluster of RandomWalker

```
class agent extends PVector {
  private int step;

  agent(int _x, int _y, int _step)
    { super(_x, _y);
    step = _step;
  }

  void
    update(){
    move();
    display();
  }

  private void move() {
    PVector rand = PVector.random2D();
    super.add(rand);
  }

  private void display() {
    strokeWeight(5);
    point(super.x, super.y);
  }
}

class cluster extends ArrayList<agent> {
  cluster() {
```

```
    super();
  }

  void add(int x, int y, int step){
    super.add(new agent(x,y,step));
  }

  void update() {
    for (int i=0; i<super.size (); i++) {
      super.get(i).update();
    }
  }
}

cluster c;

void setup() {
  size(200, 200);
  c = new cluster();
}

void draw() {
  background(255);
  c.update();
}

void mousePressed() {
  if (mouseButton == LEFT) {
    c.add(mouseX, mouseY, 5);
  }else{
    c.remove(0);
  }
```

```
}
```

PYTHON ADVANCED INTRO #02 | OBJECT-ORIENTED PROGRAMMING

Contents

OOP in GhPython

Realtime update

```
import ghpythonlib.components as gh
import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import random as rand

class RandomWalker:

 def__init__(self, step):
  self.pos = rg.Point3d(0,0,0)
  self.step = step

 def walk(self):
  sph = gh.Sphere(gh.XYPlane([0,0,0]),self.step)
  randVec = gh.PopulateGeometry(sph, 1, rand.randint(0,100))
```

```
      self.pos = rs.VectorAdd(self.pos,randVec)

a = RandomWalker(x)

```

Practice

Cluster of Random Walkers

```
import ghpythonlib.components as gh
import random as rand


class agent:
    def_init_(self, pos, step): self.pos
        = pos
        self.step = step


    def walk(self):
        sph = gh.Sphere(gh.XYPlane([0,0,0]),self.step)
        randVec = gh.PopulateGeometry(sph, 1, rand.randint(0,100))
        self.pos = gh.Addition(self.pos,randVec)


class cluster:
    def_init_(self,pts):
        self.agents = []
        for pt in pts:
            a = agent(pt,5)
            self.agents.append(a)


def update(self):
    for agent in
        self.agents:
        agent.walk()


def positions(self):
```

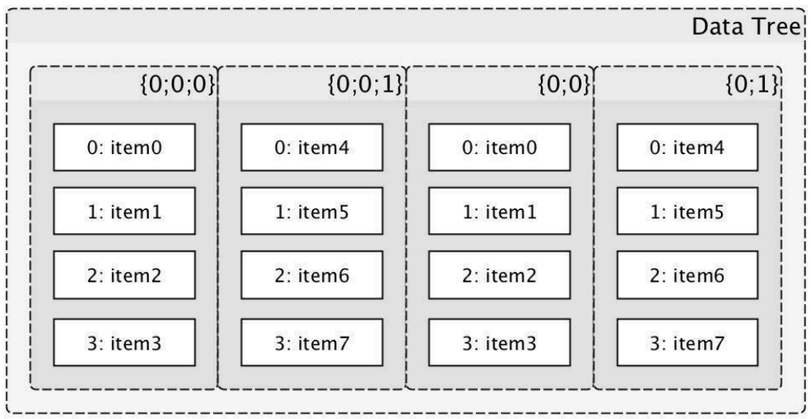
```
            pos = []
            for agent in self.agents:
                pos.append(agent.pos)
            return pos


a = cluster(x)
```

GRASSHOPPER – DATATREE

Data Tree | *Data structure*

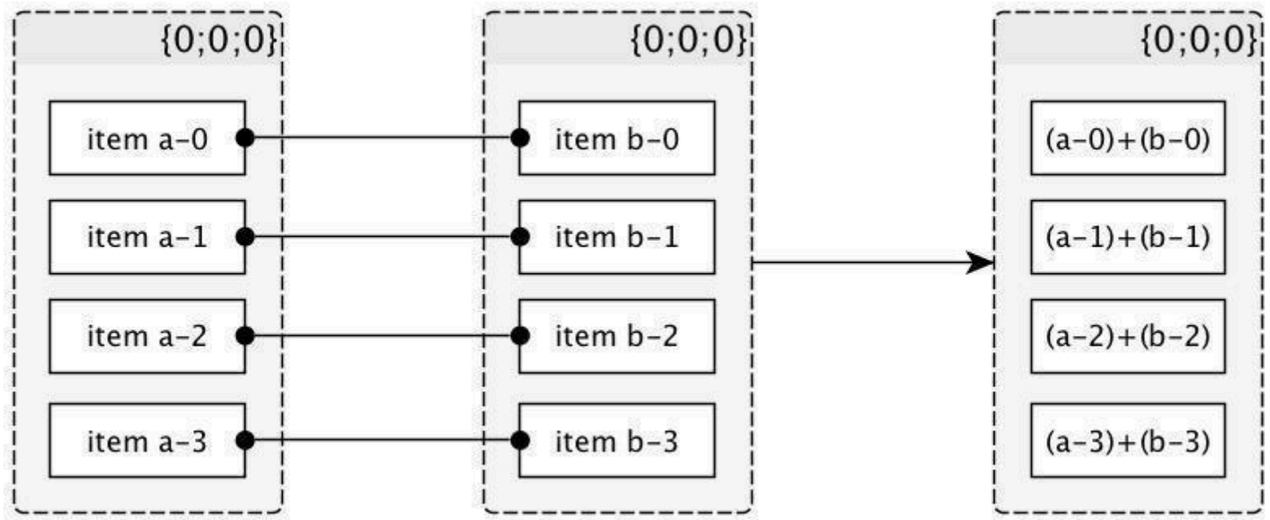
The data structures used in Grasshopper are called “**Data Tree**“.

Data Tree is composed of a bunch of lists which has a series of item (data) in it. We call the numbers of each list divided by semicolons “**Path**” and a number of item “**Index**“.

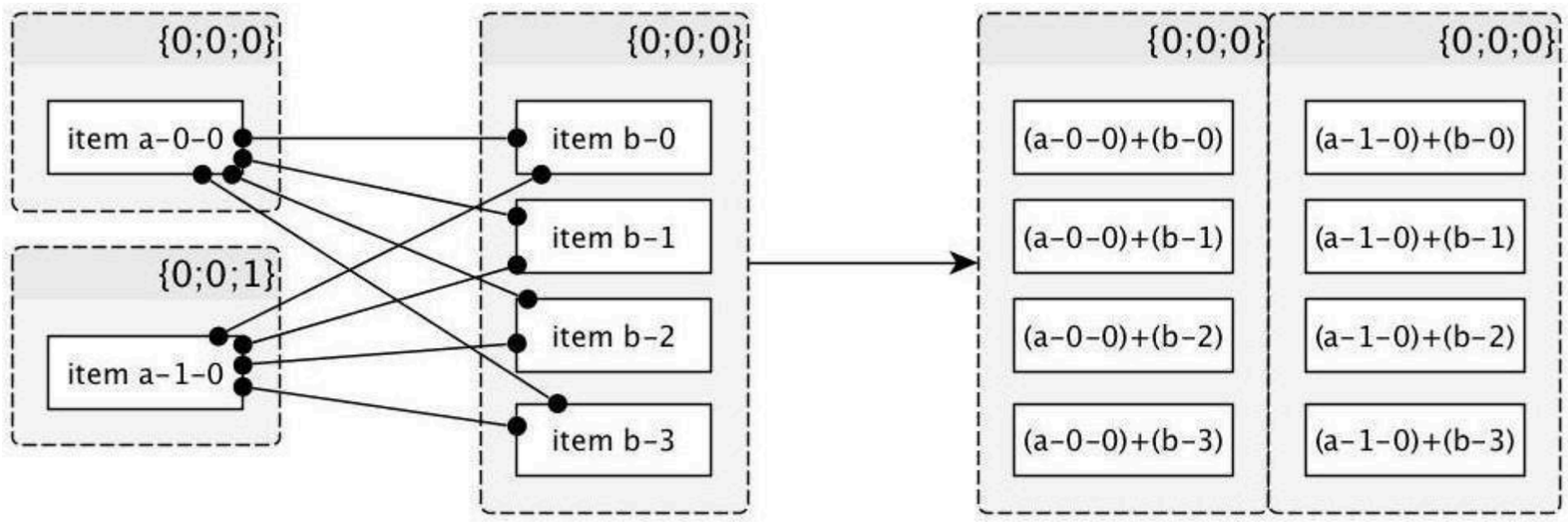


Data matching

List v.s. List matching



Data Tree v.s. Data Tree matching



Components

- Graft: put all the items in the list into each individual list

- Flatten: delete all the lists and put all the items in the lists into one list
- FlipMatrix: exchange list number (Path) and item number (Index)

Practice | *sample file*

Components

- DivideSurface
- RelItem
- Boundary