Día_000

//

Objetivos:

Introducir al alumno en un lenguaje de programación escrito, dotándolo de los conocimientos necesarios para poder afrontar el posterior aprendizaje de estructuras de código más complejas y OOP (Object-Oriented Programming). Se introduce tanto la sintaxis básica en Python como las herramientas con las que contamos en Grasshopper para su uso.

Guión:

01. import ghPython as newSkill

02. El componente GHPython. Interface y uso.

a. Entradas y salidas como variables.
b. Item, List o Tree access.
c. Type hint.
d. El editor.
e. La documentación: oficial, librerías, online.

03. Sintaxis básica.

a. Variables y tipado.
b. Operadores.
c. Importación y uso de módulos. Math, Rhinoscriptsyntax, ghpythonlib (node-in-code)…
d. **Rhinoscriptsyntax** VS Rhinocommon.
e. Condicionales y operadores booleanos.
f. Listas y bucles simples (concepto de iteración).
    i. Corte y seriación rápida.
g. Introducción a funciones.
h. Introducción a bucles dobles.

04. Ejercicios:

a. Fermat.
b. Probabilistic dead.
c. Random walker.
d. Filter_data_tree

05. Notas:

a. Floating point calculations errors.
b. Floating point arithmetic: issues and limitations.
c. Writing Good Code Is a Lot Like Making Beautiful Music.

Ej_000

```
import rhinoscriptsyntax as rs
import math as m
import ghpythonlib.components as gh

auxPts = []

for i in range(It):
    angTheta = i*Alpha
    pt = rs.Polar(ptB,angTheta,C*m.sqrt(angTheta))
    auxPts.append(pt)

Pts = auxPts
vor = gh.Voronoi(Pts,rad)
```

Ej_001

```
import rhinoscriptsyntax as rs
import random as rd

circ = []
probList = []

for c in Circles:
    center = rs.CircleCenterPoint(c)
    dist = rs.Distance(ptO,center)

    #Giving a dice to every circle and rolling it.
    r = rd.random()

    #Translating distances into 0-1 range
    prob = (dist)/dMax

    if prob>r:
        circ.append(c)
        probList.append(prob)

circlesCulled = circ
```

Ej_002

```
import Rhino.Geometry as rg
import random as r

ptList = []
ptNew = pt0
r.seed(s)

for i in range(it):

    if rg.BrepTrim.Contains(crv,pt0) == rg.PointContainment.Outside: break

    v3d = rg.Vector3d(r.randint(-1,1),r.randint(-1,1),0)
    ptNew = ptNew+v3d

    lastPt = ptNew

    while rg.BrepTrim.Contains(crv,ptNew)== rg.PointContainment.Outside:
        newv3d = rg.Vector3d(r.randint(-1,1),r.randint(-1,1),0)
        ptNew = lastPt + newv3d

    ptList.append(ptNew)

a = ptList
```

Ej_003

DivideSrf

```
import Rhino.Geometry as rg
from Grasshopper import DataTree as Tree
from Grasshopper.Kernel.Data import GH_Path as Path

ptList = []
ptTree = Tree[object]()

for i in range(uDiv+1):
    for j in range(vDiv+1):

        tempPt = srf.Evaluate(i/uDiv,j/vDiv,2)[1]
        ptTree.Add(tempPt,Path(i))

a = ptTree
```

Planarize

```
import Rhino.Geometry as rg
import rhinoscriptsyntax as rs

#Creamos un plano con tres puntos cualesquiera
plane = rg.Plane(pts[0],pts[1],pts[2])

#Proyectamos el punto 3 sobre el plano calculado
newPt3 = plane.ClosestPoint(pts[3])

#Calculamos el desplazamiento de cada punto
dev = pts[3].DistanceTo(newPt3)

#Sustituimos el punto 3 por su nuevo valor
pts[3] = newPt3
lines = []

#Creamos una marca en cada panel fuera de tolerancia
if dev>T:

    l0 = rg.Line(pts[0],pts[2])
    l1 = rg.Line(pts[1],pts[3])
    lines.extend([l0,l1])

#Creamos la polilinea de cada panel y calculamos su area
```

```
pol = rg.PolylineCurve((pts[0],pts[1],pts[2],pts[3],pts[0]))

areaObj = rg.AreaMassProperties.Compute(pol)
area = areaObj.Area

a = pts
w = lines
```