Advanced Design Scripting & Parametrics

Introduction, Grids

Class motivations

- "Computational thinking"
  - Learn the constructs of computing, programming, and AI
    Problems into systems, sequence & partition development, etc.
  - Computation geometry & geometric algorithms
- Skills
  - advanced Rhino, what's going on "under the hood", how to control it
  - Python – a very easy & powerful programming language
    an ecosystem of libraries, APIs
  - How to hack
- And, then there's AI…

Class schedule

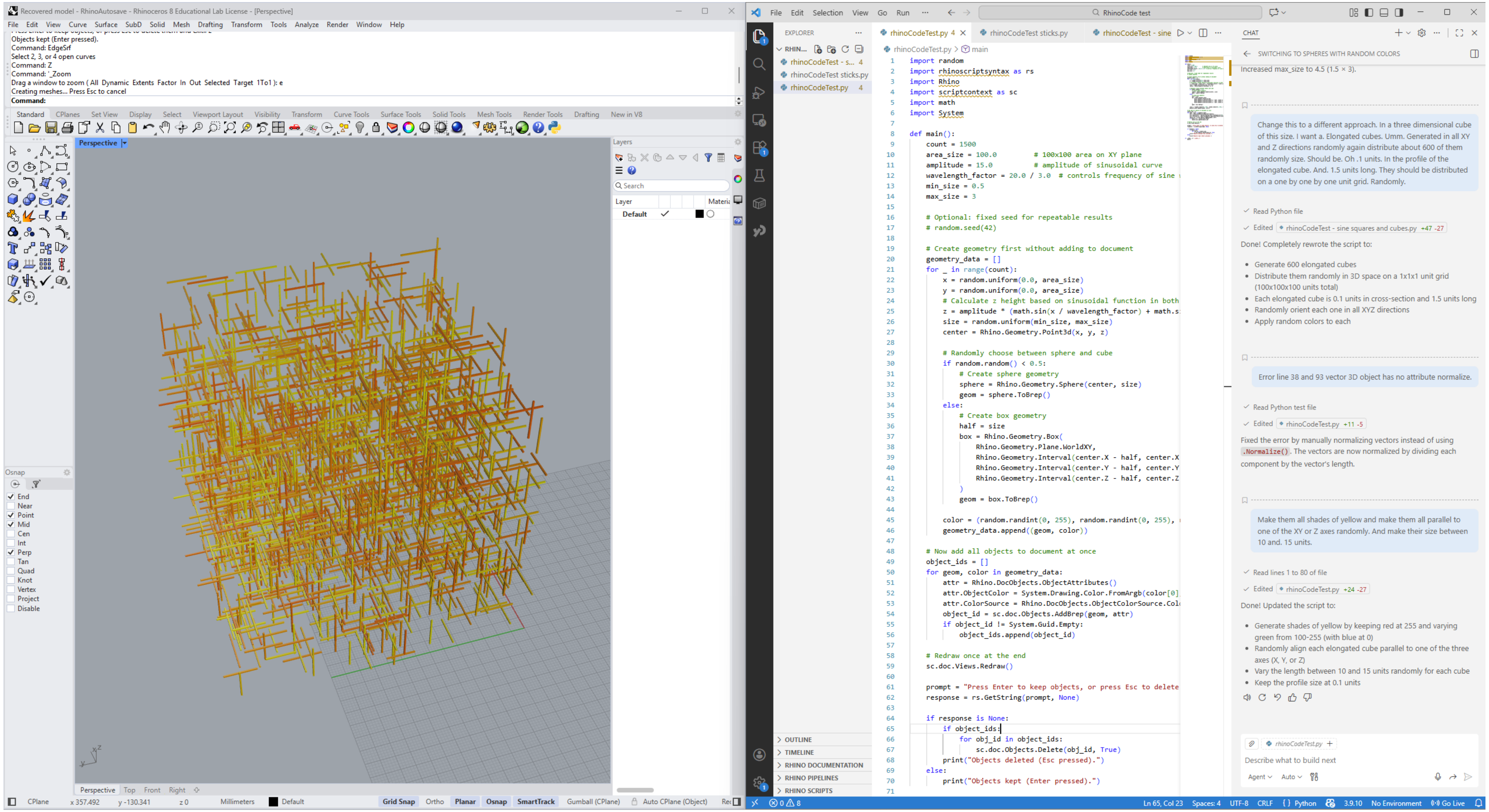
Class #	Date	Topic	Content
1	13-Jan	**Intro & Python**	Pytho online resources (W3C)
2	20-Jan	**Grids, Rhinoscript**	RhinoScript library
3	27-Jan	**Object Oriented Programming & Rhino Common**	Rhino Common Library
4	3-Feb	**Linear Algebra and Transformations**	Points, lines, planes, transforms
5	10-Feb	**Curves and Surfaces**	Curves & surfaces
6	17-Feb	No class	
7	24-Feb	**Breps & meshes**	Breps, meshes
8	3-Mar	Spring Break - no class	
9	10-Mar	**Data 1 - Tabular Data**	Tabular data, excel integration
10	17-Mar	**Data 2 - Trees & Lists**	JSON / unstructured data
11	24-Mar	**Recursion**	
12	31-Mar	**Inputs**	RhinoGet and Eto input forms
13	7-Apr	TBD	
14	14-Apr	TBD	
15	21-Apr	TBD	
16	28-Apr	TBD	
17	TBD	Final exam / presentation	

Setting up the AI coding environment

- Install
  - visual studio code https://code.visualstudio.com/
  - git https://git-scm.com/
  - github desktop https://desktop.github.com/download/

- Github registration
- Access Copilot from VS Code
- Setting up the RhinoPython "integration"

Vibe coding *new!*



https://developer.rhino3d.com/guides/rhinopython/

Rhino.Python Guides

Quickly add functionality to Rhino or automate repetitive tasks.

Overview
- What is Rhino.Python?

Getting Started
- Your First Python Script in Rhino (Windows, Mac, Grasshopper)
- Where to get help…
- Troubleshooting Installation
- Developer samples on GitHub
- Scripting discussions on Discourse

Python Editor
- ScriptEditor Command
- Canceling a Python script in Rhino
- Running a Python script in Rhino

Python in Grasshopper
- Script Component
- Creating Global Sticky Variables
- Node in Code from Python.
- Custom GhPython Baking Component
- Grasshopper data trees and Python
- GhPython Common Questions and Answers

Fundamentals
- Python Basic Syntax
- Python Procedures
- Python Data Types
- Python Variables
- Python Conditionals
- Python Looping
- Python Operators
- Python Code Conventions
- Python Dictionaries

Python in Rhino
- RhinoScriptSyntax in Python
- Points in Python
- List of Points in Python
- Colors in Python
- Lines in Python
- Vectors in Python
- Planes in Python
- Rhino objects in Python
- Point and Vector Methods
- Line and Plane Methods
- Rhino NURBS Geometry Overview

Other Resources
- Rhino Scripting Forum (Discourse)
- Rhino.Python Samples
- Rhino.Python Developer Samples GitHub
- Designalyze Python Tutorials
- Plethora Project
- Steve Baer's Blog
- Python Beginner's Guide
- Tutorials Point Python Series
- Rhino.Python Dash Docset

Rhino.Python 101

Introduction

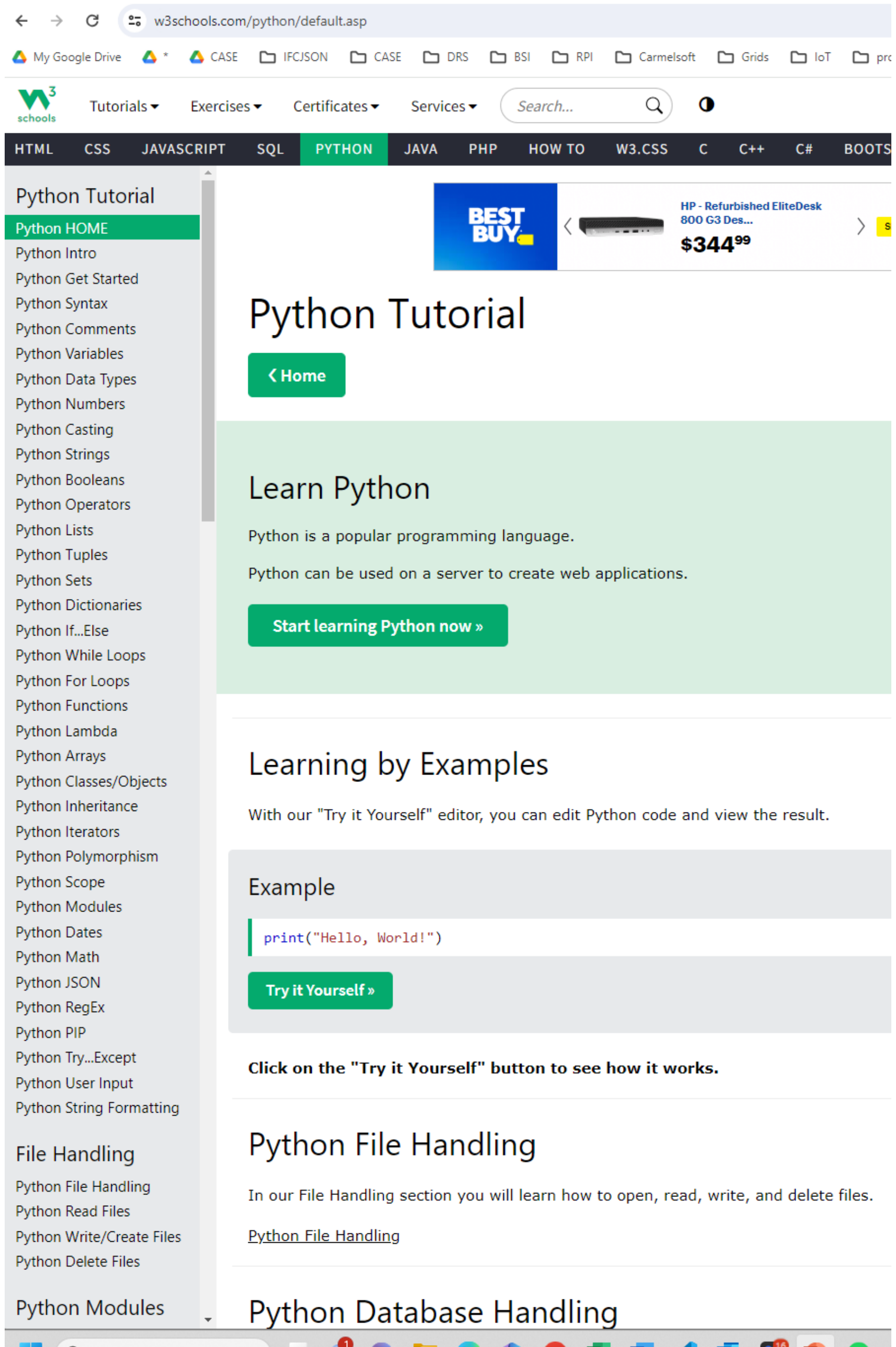
Where to find help

1. What's it all about?
2. Python Essentials
3. Script anatomy
4. Operators and functions
5. Conditional execution
6. Tuples, Lists and Dictionaries
7. Classes
8. Geometry

Download the Rhino.Python 101 Primer as a single PDF

Intermediate
- How to read and write a simple file
- Code-Driven File IO
- How to read and write a CSV files
- How to use JSON
- Using Python Dictionary as a database
- How to get user input in a script
- Creating a script and module
- APIs Available to Python
- Calling Overloaded Methods from Python
- Generating Random Numbers in Python
- Providing Arguments for By-Reference Parameters

Custom Dialogs in Eto
- Writing Custom Eto forms in Python



https://www.w3schools.com/python/default.asp developer.rhino3d.com/api/RhinoScriptSyntax/

My Google Drive CASE IFCJSON CASE DRS BSI RPI Carmelsoft Grids IoT projects EBESS All Bookmarks

RhinoDeveloper Guides Samples API Videos Community sign in

RhinoScriptSyntax

Search

- application
- block
- curve
- dimension
- document
- geometry
- grips
- group
- hatch
- layer
- light
- line
- linetype
- material
- mesh
- object
- plane
- pointvector
- selection
- surface
- toolbar
- transformation
- userdata
- userinterface
- utility
- view

API References /

RhinoScriptSyntax

application

AddAlias
AddSearchPath
AliasCount
AliasMacro
AliasNames
AppearanceColor
AutosaveFile
AutosaveInterval
BuildDate
ClearCommandHistory
Command
CommandHistory
DefaultRenderer
DeleteAlias
DeleteSearchPath
DisplayOleAlerts
EdgeAnalysisColor
EdgeAnalysisMode
EnableAutosave
EnablePlugIn
ExeFolder
ExePlatform
ExeServiceRelease
ExeVersion
Exit
FindFile
GetPlugInObject
InCommand
InstallFolder
IsAlias
IsCommand
IsPlugIn
IsRunningOnWindows
LastCommandName
LastCommandResult
LocaleID
Ortho
Osnap
OsnapDialog
OsnapMode
Planar

https://developer.rhino3d.com/api/RhinoScriptSyntax/

Setting up the AI coding environment

- Install
  - visual studio code https://code.visualstudio.com/
  - git https://git-scm.com/
  - github desktop https://desktop.github.com/download/

- Github registration, email me your github ID
- Github course repository: https://github.com/Drshelden/DP2
- Access Copilot from VS Code
- Setting up the RhinoPython “integration”

RhinoPython integration: Create an Alias

1. Tools>Options>Aliases, Import “MakeDPAlias.txt”
2. Run DP2setup.bat
3. In Rhino: “Toolbar”, File->Open, “Design Programming 2.rui”
4. Open the folder “"C:_LOCAL\DP2\RhinoCode“ in VS Code

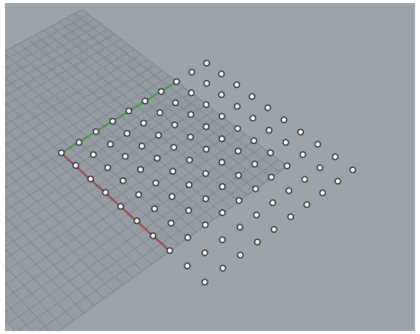
In Rhino:

- Test the DP alias and the Design Programming button.
- Save "C:_LOCAL\DP2\RhinoCode\WorkingCode.py“ and DP again

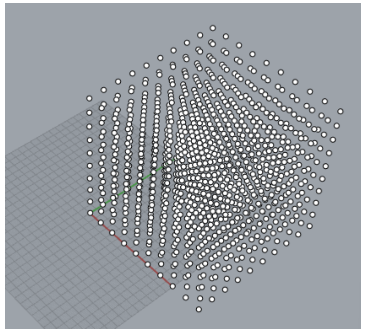
Basic importing of libraries

Always start your program with this:

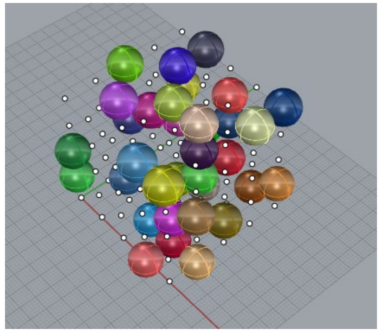
```python
import rhinoscriptsyntax as rs
import scriptcontext as sc

import System
import System.Collections.Generic
import Rhino
import random
import math
```



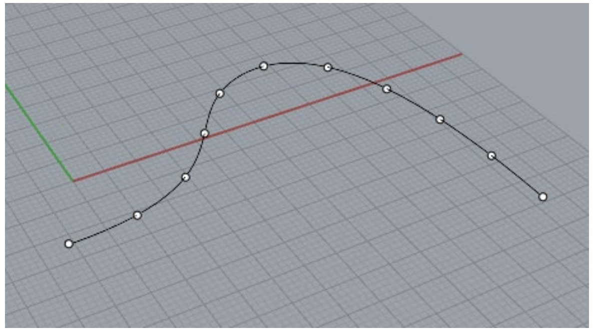
2D Grid



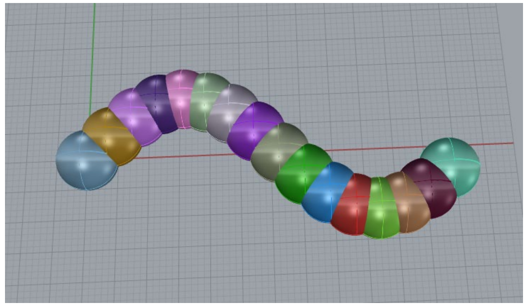
3D Grid



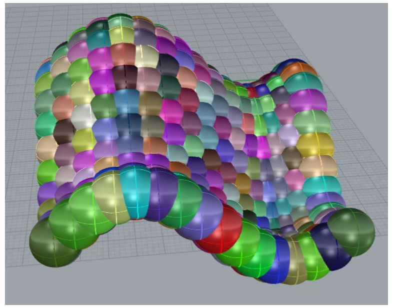
3D Grid with Random Spheres



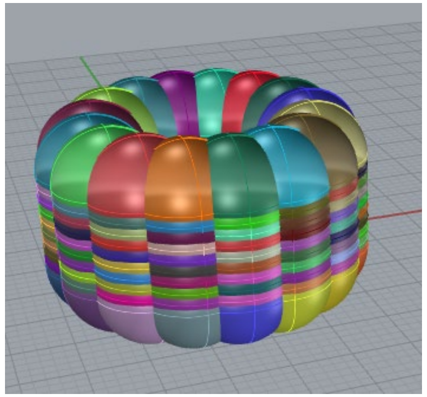
along curve



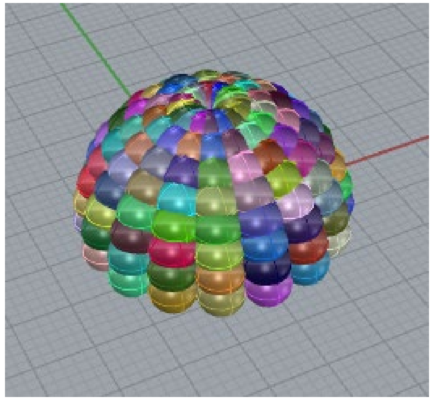
Sine Function 2D



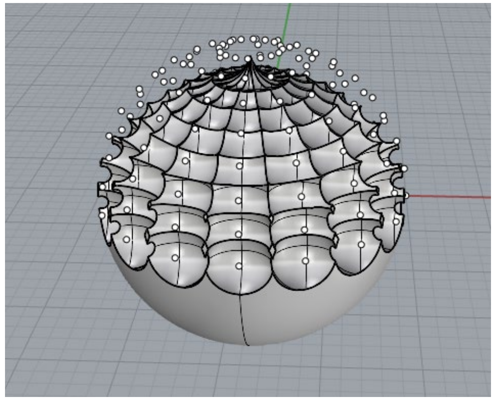
Sine Function 3D



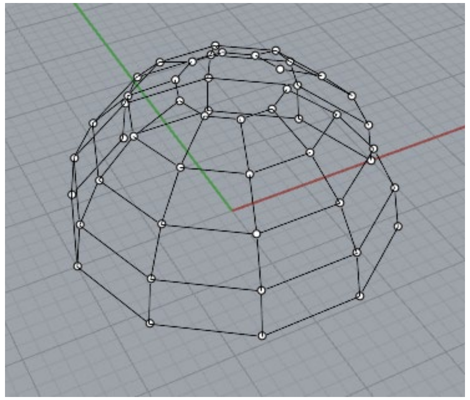
cylindrical coordinates



spherical coordinates



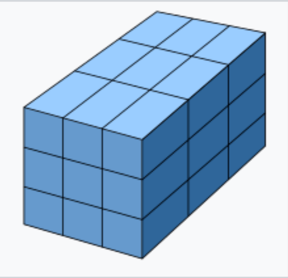
spherical coordinates with boolean



spherical coordinates with grid

What’s a “grid”?

A **regular grid** is a tessellation of *n*-dimensional Euclidean space by congruent parallelotopes (e.g. bricks).[1] Its opposite is irregular grid.



Example of a regular grid

Grids of this type appear on graph paper and may be used in finite element analysis, finite volume methods, finite difference methods, and in general for discretization of parameter spaces. Since the derivatives of field variables can be conveniently expressed as finite differences,[2] structured grids mainly appear in finite difference methods. Unstructured grids offer more flexibility than structured grids and hence are very useful in finite element and finite volume methods.

Each cell in the grid can be addressed by index (i, j) in two dimensions or (i, j, k) in three dimensions, and each vertex has coordinates $(i \cdot dx, j \cdot dy)$ in 2D or $(i \cdot dx, j \cdot dy, k \cdot dz)$ in 3D for some real numbers *dx*, *dy*, and *dz* representing the grid spacing.

Related grids [edit]

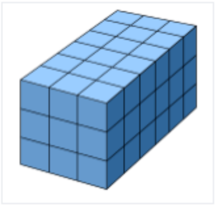
A **Cartesian grid** is a special case where the elements are unit squares or unit cubes, and the vertices are points on the integer lattice.

A **rectilinear grid** is a tessellation by rectangles or rectangular cuboids (also known as rectangular parallelepipeds) that are not, in general, all congruent to each other. The cells may still be indexed by integers as above, but the mapping from indexes to vertex coordinates is less uniform than in a regular grid. An example of a rectilinear grid that is not regular appears on logarithmic scale graph paper.

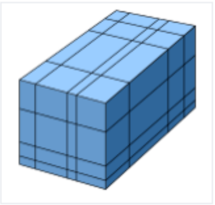
A **skewed grid** is a tessellation of parallelograms or parallelepipeds. (If the unit lengths are all equal, it is a tessellation of rhombi or rhombohedra.)

A **curvilinear grid** or **structured grid** is a grid with the same combinatorial structure as a regular grid, in which the cells are quadrilaterals or [general] cuboids, rather than rectangles or rectangular cuboids.

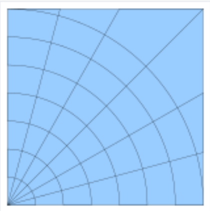
Examples of various grids



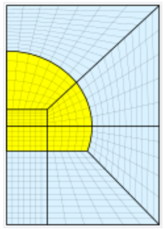
3-D Cartesian grid



3-D rectilinear grid



2-D curvilinear grid



Non-curvilinear combination of different 2-D curvilinear grids

Starter file for our class: Class01.00 Starter.py

```python
#! python3
"""Testing pip install specific packages"""
# r: numpy

import rhinoscriptsyntax as rs
import scriptcontext as sc

import System
import System.Collections.Generic
import Rhino
import random
import math
import numpy as np

xCount = 5
yCount = 5
zCount = 5
xScale = 10
yScale = 10
zScale = 10
rSphere = 5


# Your Code Here!


 # done!
```

#some instructions to python

imports a namespace and calls it by the nickname “rs”

#import some libraries of code – just like this code

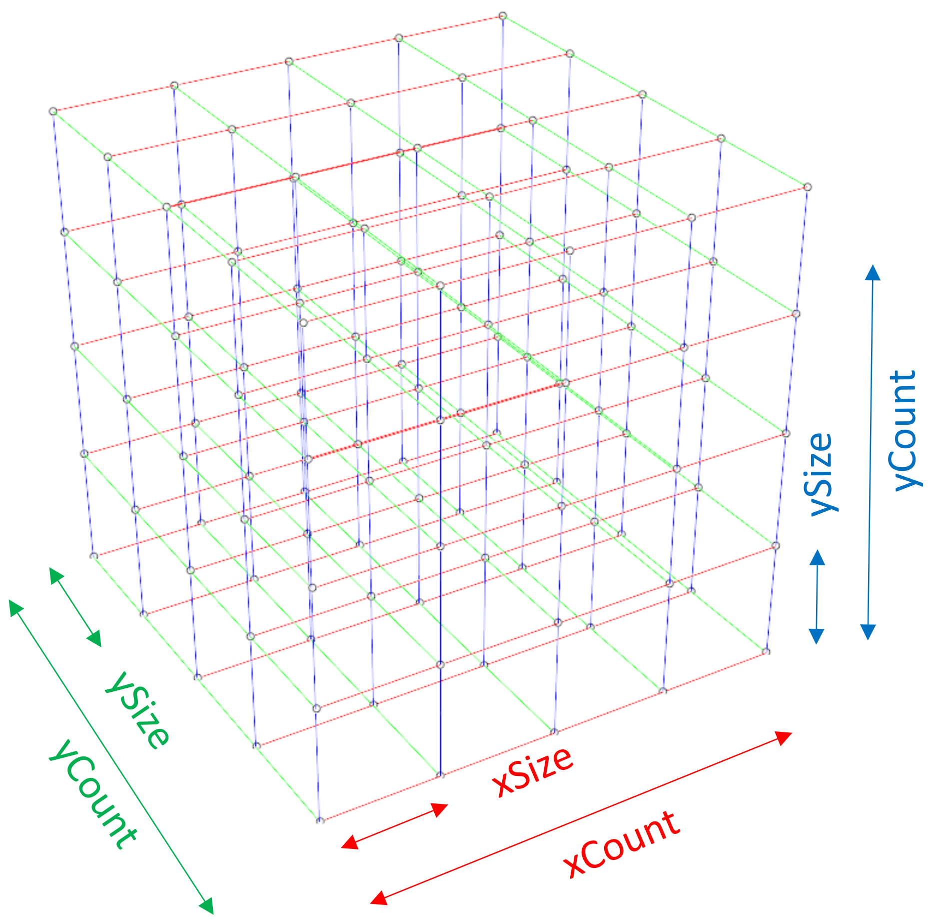
#define some variables and set their initial values

#comments – ignored by Python but useful for humans

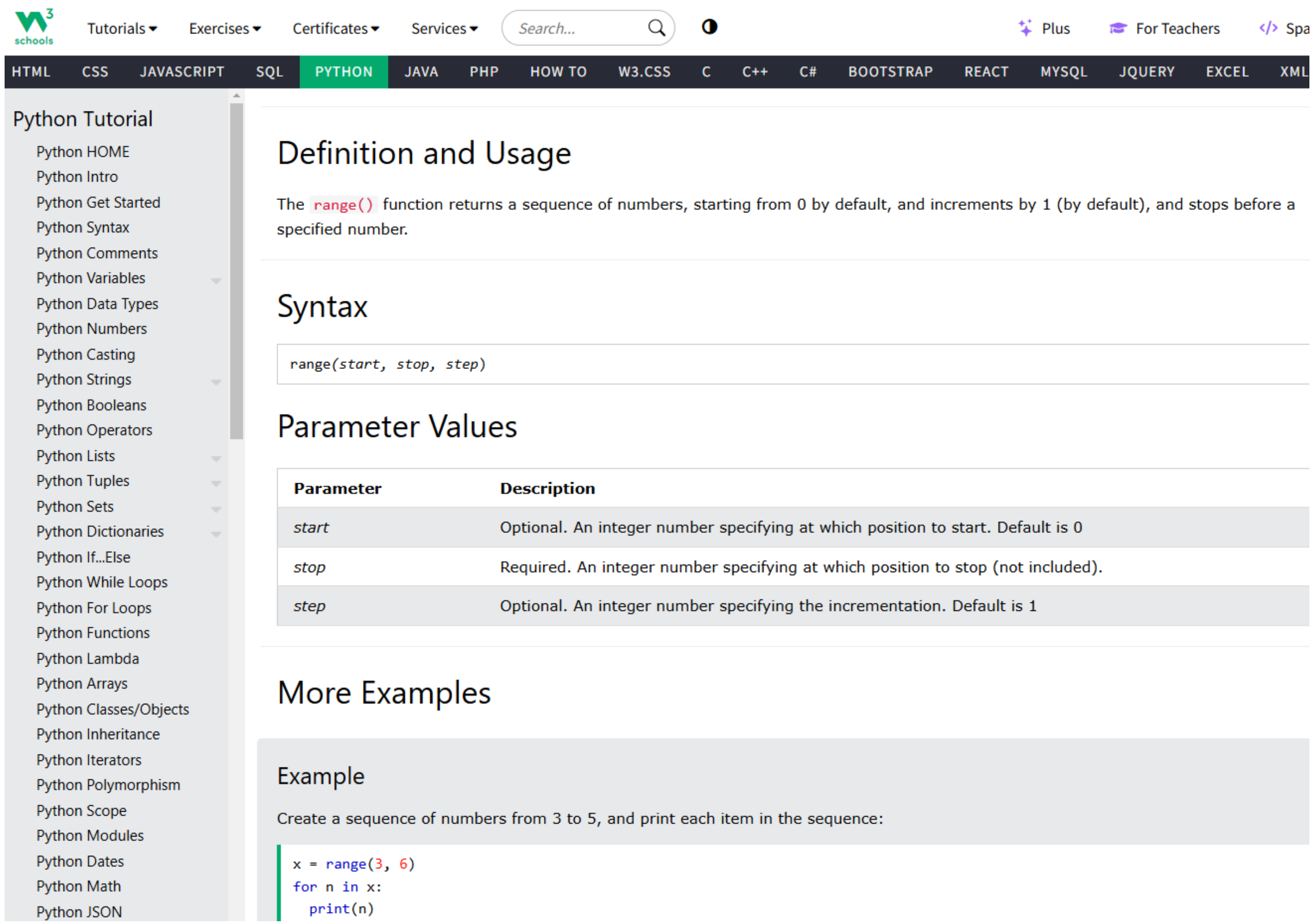
Starter file for our class: Class01.00 Starter.py

```python
#! python3
"""Testing pip install specific packages"""
# r: numpy

import rhinoscriptsyntax as rs
import scriptcontext as sc

import System
import System.Collections.Generic
import Rhino
import random
import math
import numpy as np

xCount = 5
yCount = 5
zCount = 5
xScale = 10
yScale = 10
zScale = 10
rSphere = 5


# Your Code Here!
```

```python
for x in range(xCount):
    print(x)
```



range() # python function



Tutorials ▾ Exercises ▾ Certificates ▾ Services ▾ Search... Plus For Teachers Spa

HTML CSS JAVASCRIPT SQL PYTHON JAVA PHP HOW TO W3.CSS C C++ C# BOOTSTRAP REACT MYSQL JQUERY EXCEL XML

Python Tutorial

Python HOME
Python Intro
Python Get Started
Python Syntax
Python Comments
Python Variables
Python Data Types
Python Numbers
Python Casting
Python Strings
Python Booleans
Python Operators
Python Lists
Python Tuples
Python Sets
Python Dictionaries
Python If...Else
Python While Loops
Python For Loops
Python Functions
Python Lambda
Python Arrays
Python Classes/Objects
Python Inheritance
Python Iterators
Python Polymorphism
Python Scope
Python Modules
Python Dates
Python Math
Python JSON

Definition and Usage

The `range()` function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

Syntax

```
range(start, stop, step)
```

Parameter Values

Parameter	Description
start	Optional. An integer number specifying at which position to start. Default is 0
stop	Required. An integer number specifying at which position to stop (not included).
step	Optional. An integer number specifying the incrementation. Default is 1

More Examples

Example

Create a sequence of numbers from 3 to 5, and print each item in the sequence:

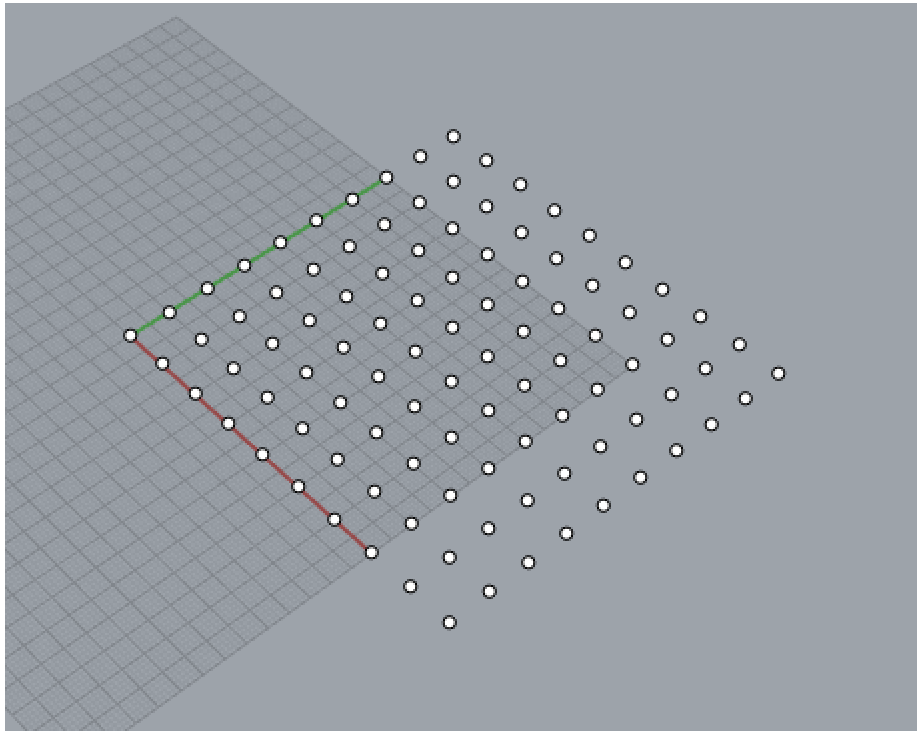
```
x = range(3, 6)
for n in x:
  print(n)
```

A predefined **function** in the python base language library:

Given an integer *stop*
Returns an **array** of integers
[0, 1, … *stop* – 1]

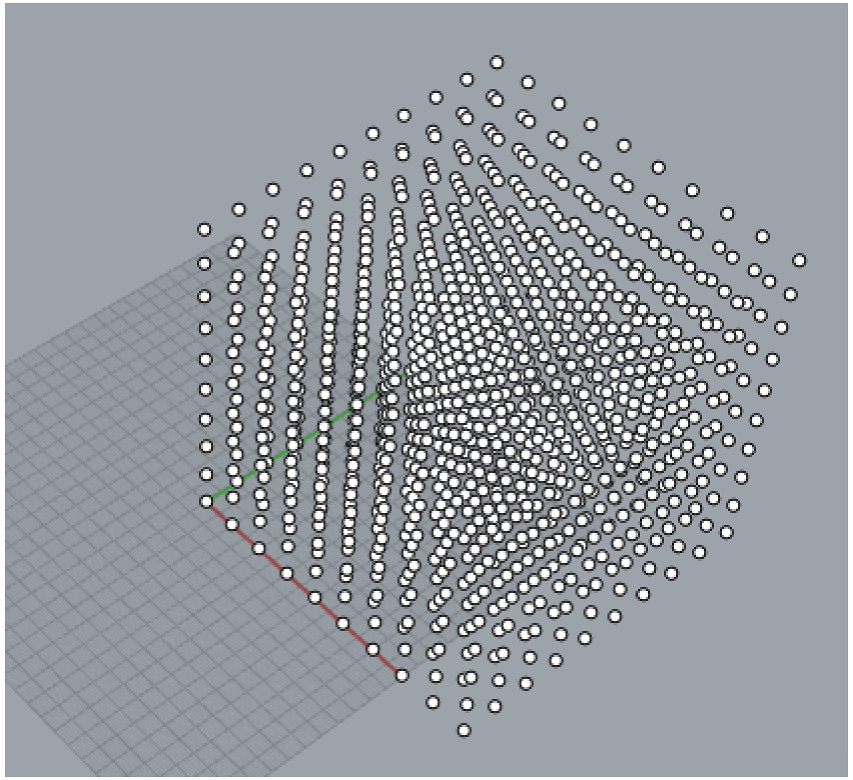
Some options:
range(*stop*) -> [0, …, *stop* – 1]
range(*start*, *stop*) -> [*start*, …, *stop* -1]
range(*start*, *stop*, *step*) ->
[*start*, *start* + *step*, *start* + 2**step*, …, *stop* -1]

x = range(5) # creates an array [0,…,4]
for **n** in range(x): #makes an arbitrary variable n for each value in x assigns this value to n and runs the nested code
print(n)



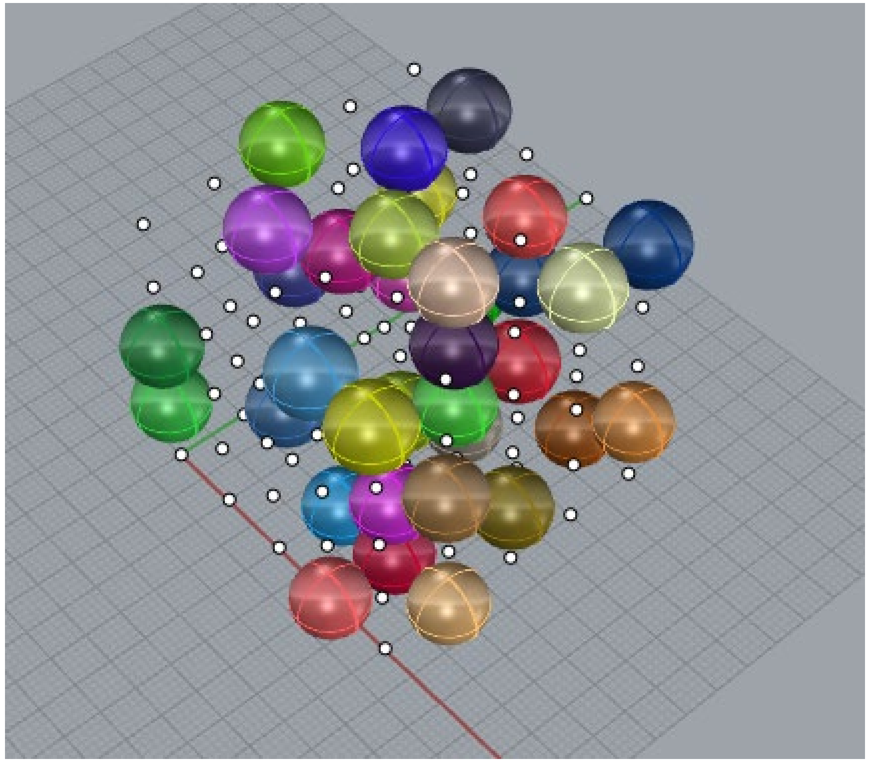
Class01.01 2D Grid.py

```
#! python3
"""Testing pip install specific packages"""
# r: numpy

import rhinoscriptsyntax as rs
import scriptcontext as sc

import System
import System.Collections.Generic
import Rhino
import random
import math
import numpy as np

xCount = 10
yCount = 10
zCount = 10
xScale = 10
yScale = 10
zScale = 10
rSphere = 5


# Print out the coordinates of each point
for x in range(xCount):
    for y in range(yCount):
                pointId = rs.AddPoint((x * xScale, y * yScale,0))
                print(x, y, pointId)

 # done!
```



Class01.02 3D Grid.py

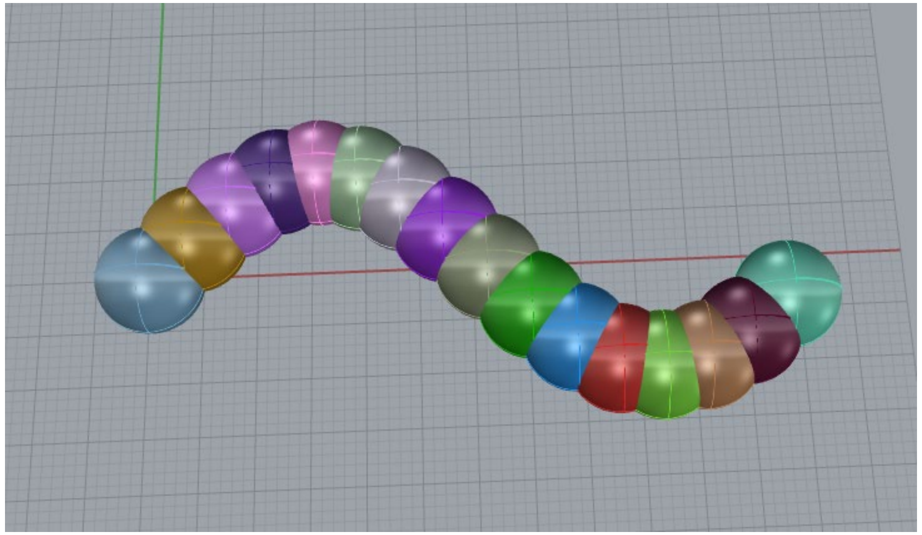
```
#! python3
"""Testing pip install specific packages"""
# r: numpy

import rhinoscriptsyntax as rs
import scriptcontext as sc

import System
import System.Collections.Generic
import Rhino
import random
import math
import numpy as np

xCount = 10
yCount = 10
zCount = 10
xScale = 10
yScale = 10
zScale = 10
rSphere = 5


# Print out the coordinates of each point
for x in range(xCount):
    for y in range(yCount):
            for z in range(zCount):
                pointId = rs.AddPoint((x * xScale, y * yScale,z * zScale))
                print(x, y, pointId)

 # done!
```



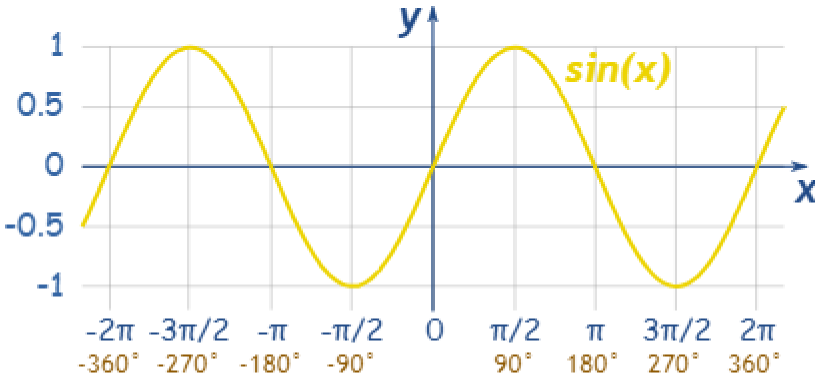
Class01.03 3D Grid with Random Spheres.py

random and random.random()

```
#! python3
"""Testing pip install specific packages"""
# r: numpy

import rhinoscriptsyntax as rs
import scriptcontext as sc

import System
import System.Collections.Generic
import Rhino
import random
import math
import numpy as np

xCount = 5
yCount = 5
zCount = 5
xScale = 10
yScale = 10
zScale = 10
rSphere = 5


# Print out the coordinates of each point
for x in range(xCount):
    for y in range(yCount):
            for z in range(zCount):
                pointId = rs.AddPoint((x * xScale, y * yScale,z * zScale))
                print(x, y, pointId)

                if (random.random() < 0.25):
                    sphereId= rs.AddSphere((x * xScale, y * yScale,z * zScale), rSphere)
                    rs.ObjectColor(sphereId, [int(random.random() * 255), int(random.random() * 255), \
                    int(random.random() * 255) ])

 # done!
```



Class01.05 Sine Function 2D.py

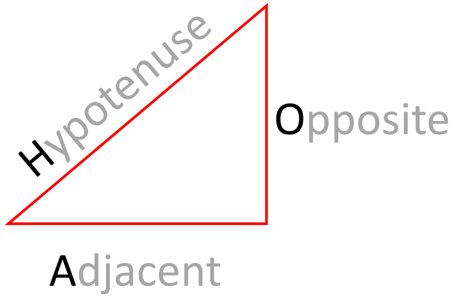



```
#! python3
"""Testing pip install specific packages"""
# r: numpy

import rhinoscriptsyntax as rs
import scriptcontext as sc

import System
import System.Collections.Generic
import Rhino
import random
import math
import numpy as np


xCount = 16
yCount = 5
zCount = 5
xScale = 10
yScale = 10
zScale = 10
rSphere = 5

listValues = np.arange(0, 2 * math.pi, 2 * math.pi / xCount)
print(listValues)

# Print out the coordinates of each point
for x in listValues:
        y = math.sin(x)
        sphereId = rs.AddSphere( (x * xScale, y * yScale, 0), rSphere )
        #print(sphereId)
        rs.ObjectColor(sphereId, [int(random.random() * 255), int(random.random() * 255), \
        int(random.random() * 255) ])

 # done!
```

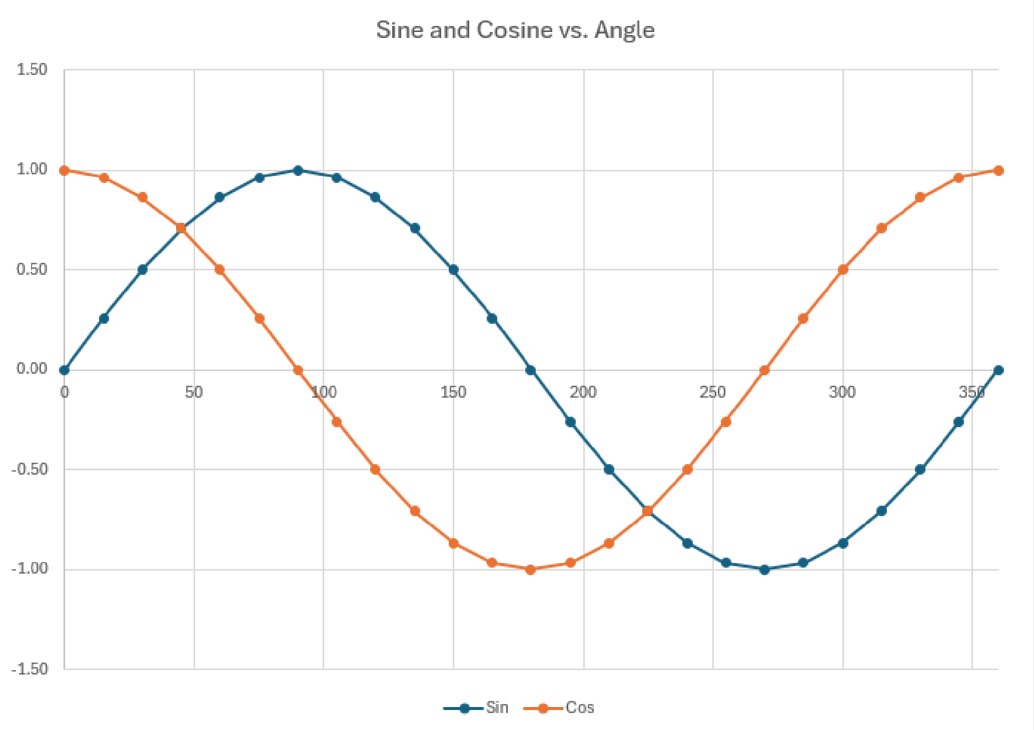
$\text{Sin} = \dfrac{O}{H}$

$\text{Cos} = \dfrac{A}{H}$

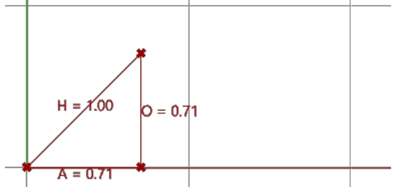
$\text{Tan} = \dfrac{O}{A}$



Degrees		Radians	Sin	Cos
0	0	0.00	0.00	1.00
15		0.26	0.26	0.97
30		0.52	0.50	0.87
45		0.79	0.71	0.71
60		1.05	0.87	0.50
75		1.31	0.97	0.26
90	π / 2	1.57	1.00	0.00
105		1.83	0.97	-0.26
120		2.09	0.87	-0.50
135		2.36	0.71	-0.71
150		2.62	0.50	-0.87
165		2.88	0.26	-0.97
180	π	3.14	0.00	-1.00
195		3.40	-0.26	-0.97
210		3.67	-0.50	-0.87
225		3.93	-0.71	-0.71
240		4.19	-0.87	-0.50
255		4.45	-0.97	-0.26
270	3/2 π	4.71	-1.00	0.00
285		4.97	-0.97	0.26
300		5.24	-0.87	0.50
315		5.50	-0.71	0.71
330		5.76	-0.50	0.87
345		6.02	-0.26	0.97
360	2 π	6.28	0.00	1.00



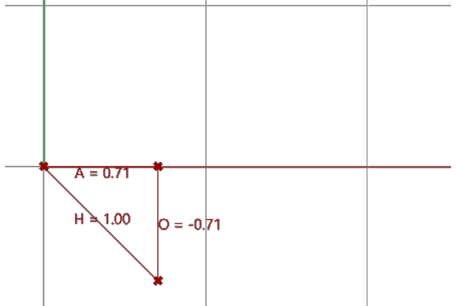
45.0 deg = 0.79 radians
Sin = O/H = 0.71 / 1.0 = 0.71
Cos = A/H = 0.71 / 1.0 = 0.71
Tan = O/A = 0.71 / 0.71 = 1.00



135.0 deg = 2.36 radians
Sin = O/H = 0.71 / 1.0 = 0.71
Cos = A/H = -0.71 / 1.0 = -0.71
Tan = O/A = 0.71 / -0.71 = -1.00

225.0 deg = 3.93 radians
Sin = O/H = -0.71 / 1.0 = -0.71
Cos = A/H = -0.71 / 1.0 = -0.71
Tan = O/A = -0.71 / -0.71 = 1.00

315.0 deg = 5.50 radians
Sin = O/H = -0.71 / 1.0 = -0.71
Cos = A/H = 0.71 / 1.0 = 0.71
Tan = O/A = -0.71 / 0.71 = -1.00



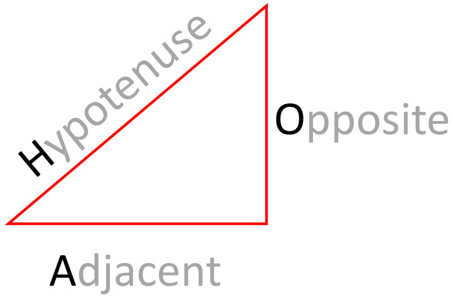
Quad	sin	cos
1	+	+
2	+	-
3	-	-
4	-	+

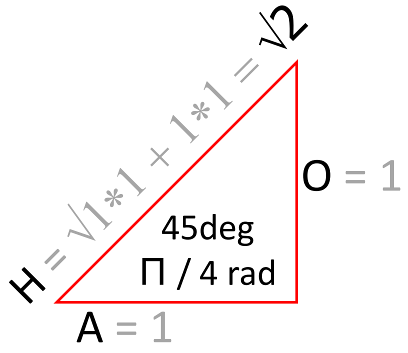
Trig to remember

$\text{Sin} = \frac{O}{H}$

$\text{Cos} = \frac{A}{H}$

$\text{Tan} = \frac{O}{A}$



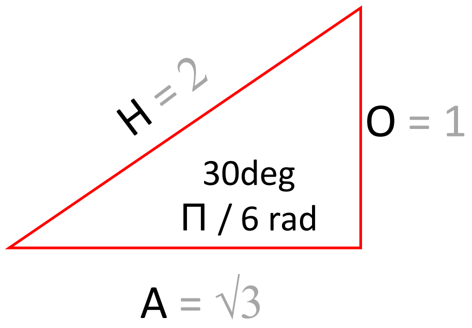


$\text{Sin} = \frac{1}{\sqrt{2}} \sim 0.707$

$\text{Cos} = \frac{1}{\sqrt{2}}$

$\text{Tan} = \frac{1}{1}$

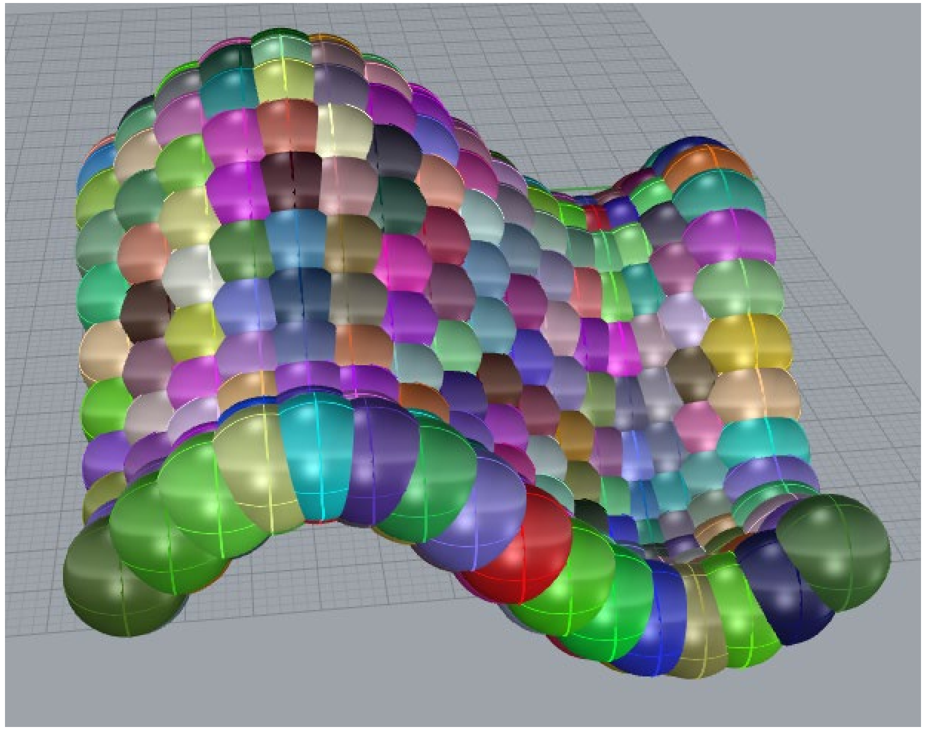
	sin	cos
0	0	1
90	1	0
180	0	-1
270	-1	0



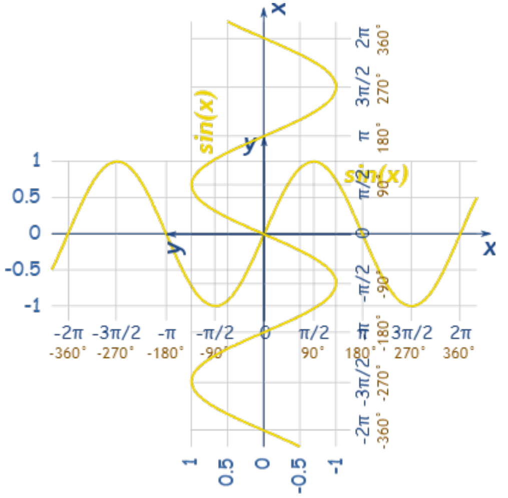
$\text{Sin} = \frac{1}{2}$

$\text{Cos} = \frac{\sqrt{3}}{2}$

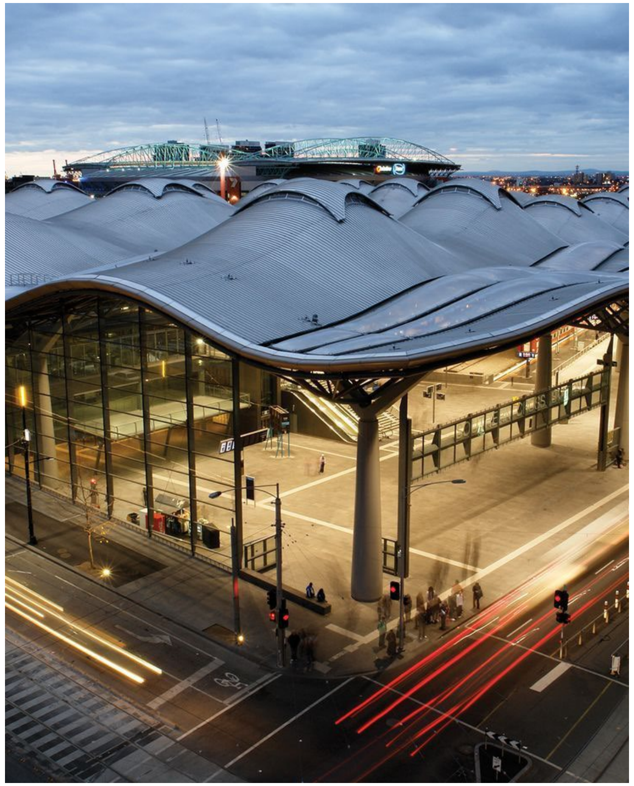
$\text{Tan} = \frac{1}{\sqrt{3}}$

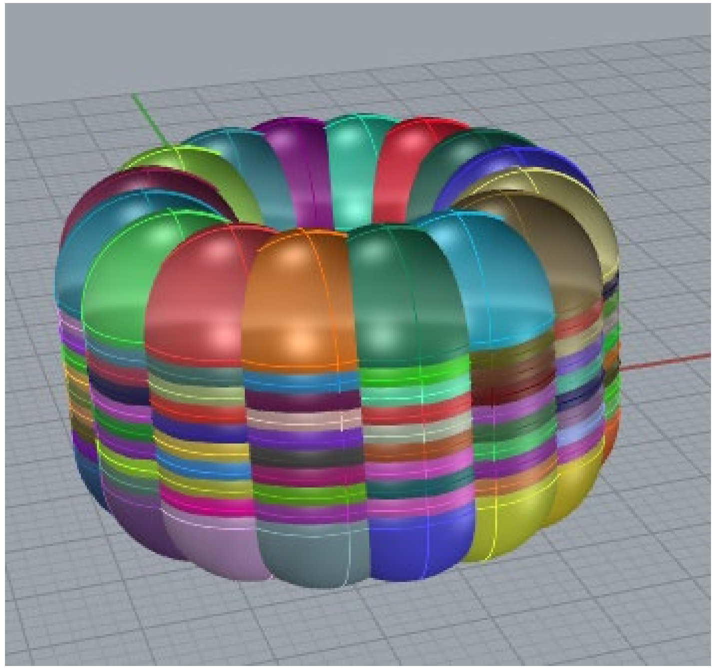


sine Function 2D.py

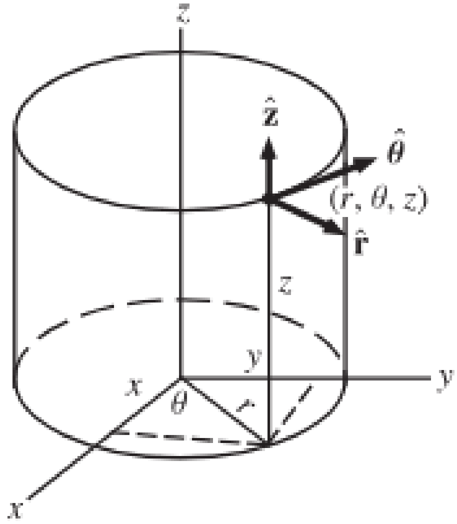



```
#! python3
"""Testing pip install specific packages"""
# r: numpy

import rhinoscriptsyntax as rs
import scriptcontext as sc

import System
import System.Collections.Generic
import Rhino
import random
import math
import numpy as np


xCount = 16
yCount = 5
zCount = 5
xScale = 10
yScale = 10
zScale = 10
rSphere = 5

listValues = np.arange(0, 2 * math.pi, 2 * math.pi / xCount)
print(listValues)

# Print out the coordinates of each point
for x in listValues:
    for y in listValues:
        z = math.sin(x) + math.sin(y)
        sphereId = rs.AddSphere( (x * xScale, y * yScale, z * zScale), rSphere )
        #print(sphereId)
        rs.ObjectColor(sphereId, [int(random.random() * 255), int(random.random() * 255), \
        int(random.random() * 255) ])

 # done!
```





Class01.06 cylindrical coordinates.py



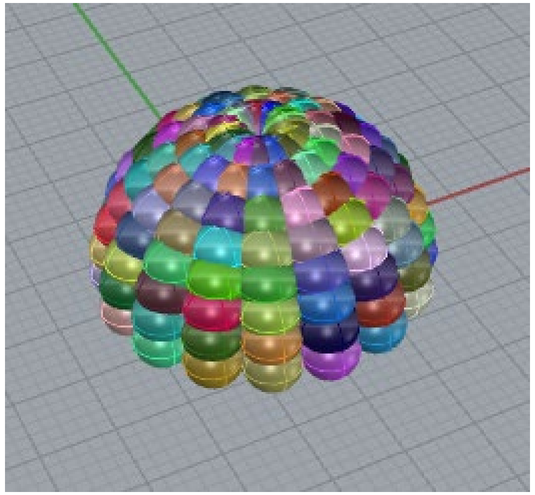
alpha

theta

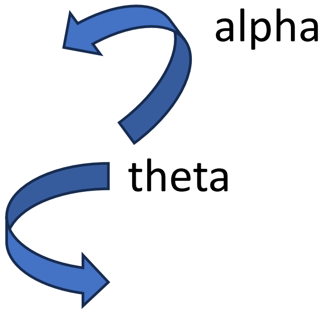
x = cos(theta)

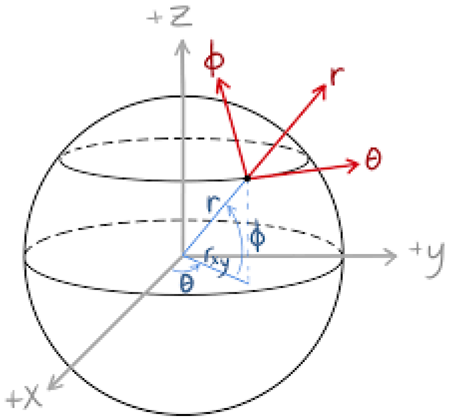
y = sin(theta)

```python
#! python3
"""Testing pip install specific packages"""
# r: numpy

import rhinoscriptsyntax as rs
import scriptcontext as sc

import System
import System.Collections.Generic
import Rhino
import random
import math
import numpy as np


xCount = 16
yCount = 5
zCount = 5
xScale = 10
yScale = 10
zScale = 10
rSphere = 5

thetaValues = np.arange(0, 2 * math.pi, 2 * math.pi / xCount) # 0 to 2PI in xCount steps
alphaValues = np.arange(0, 1.0, 1 / yCount) # 0 to PI / 2 in yCount steps


# Print out the coordinates of each point
for theta in thetaValues:
    for alpha in alphaValues:

        x = math.cos(theta)
        y = math.sin(theta)
        z = alpha

        sphereId = rs.AddSphere( (x * xScale, y * yScale, z * zScale), rSphere )
        #print(sphereId)
        rs.ObjectColor(sphereId, [int(random.random() * 255), int(random.random() * 255), \
        int(random.random() * 255) ])

# done!
```



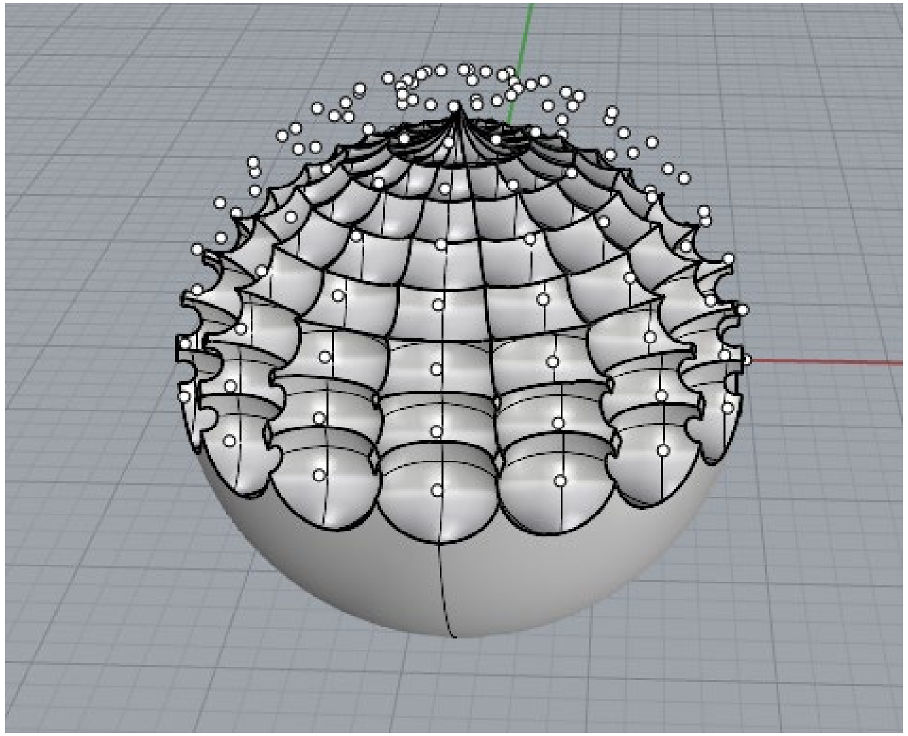
Class01.07 spherical coordinates.py





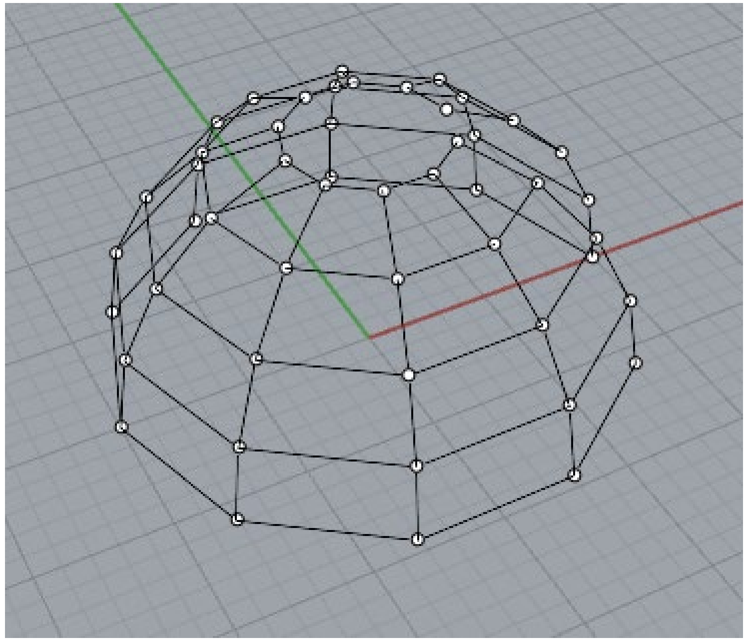
x = cos(theta) * cos(alpha)
y = sin(theta) * cos(alpha)
Z = sin(alpha)

```python
#! python3
"""Testing pip install specific packages"""
# r: numpy

import rhinoscriptsyntax as rs
import scriptcontext as sc

import System
import System.Collections.Generic
import Rhino
import random
import math
import numpy as np


xCount = 5
yCount = 5
zCount = 5
xScale = 10
yScale = 10
zScale = 10
rSphere = 2

thetaValues = np.arange(0, 2 * math.pi, 2 * math.pi /16) # 0 to 2PI in 16 steps
alphaValues = np.arange(0, math.pi / 2.0, math.pi / 16) # 0 to 1 in 0.1 steps


# Print out the coordinates of each point
for theta in thetaValues:
    for alpha in alphaValues:

        x = math.cos(theta) * math.cos(alpha)
        y = math.sin(theta) * math.cos(alpha)
        z = math.sin(alpha)


        sphereId = rs.AddSphere( (x * xScale, y * yScale, z * zScale), rSphere )
        #print(sphereId)
        rs.ObjectColor(sphereId, [int(random.random() * 255), int(random.random() * 255),
        int(random.random() * 255) ])

# done!
```



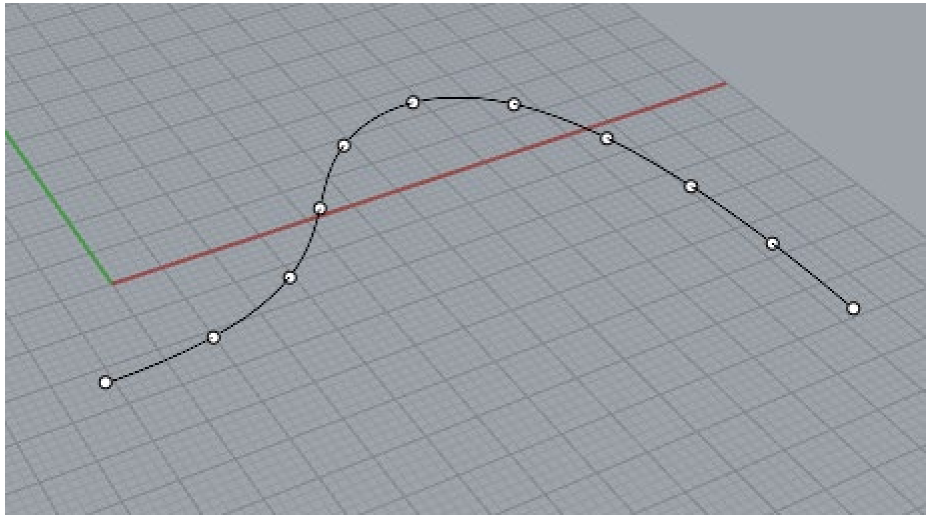
Class01.08 spherical coordinates with boolean.py

```python
#! python3
"""Testing pip install specific packages"""
# r: numpy

import rhinoscriptsyntax as rs
import scriptcontext as sc

import System
import System.Collections.Generic
import Rhino
import random
import math
import numpy as np


xScale = 10
yScale = 10
zScale = 10
rSphere = 5

thetaValues = np.arange(0, 2 * math.pi, 2 * math.pi / 16) # 0 to 2PI in 16 steps
alphaValues = np.arange(0, 1.0, 0.1) # 0 to 1 in 0.1 steps


# Print out the coordinates of each point
for theta in thetaValues:
    for alpha in alphaValues:

        x = math.cos(theta)
        y = math.sin(theta)
        z = alpha


        sphereId = rs.AddSphere( (x * xScale, y * yScale, z * zScale), rSphere )
        #print(sphereId)
        rs.ObjectColor(sphereId, [int(random.random() * 255), int(random.random() * 255), \
        int(random.random() * 255) ])

 # done!
```



Class01.09 spherical coordinates with grid.py

```
thetaValues = np.arange(0, 2 * math.pi, 2 * math.pi /xCount) # 0 to 2PI in xCount steps
alphaValues = np.arange(0, math.pi / 2.0, math.pi / yCount) # 0 to PI / 2 in yCount steps

pointIds = []


# Print out the coordinates of each point
i = 0

pointIds = [[0 for _ in range(xCount)] for _ in range(xCount)]

for theta in thetaValues:
    j = 0
    for alpha in alphaValues:

        x = math.cos(theta) * math.cos(alpha)
        y = math.sin(theta) * math.cos(alpha)
        z = math.sin(alpha)

        pointId = rs.AddPoint( (x * xScale, y * yScale, z * zScale) )
        pointIds[i][j] = pointId
        if (i > 0): rs.AddLine(pointIds[i-1][j], pointIds[i][j])
        if (j > 0): rs.AddLine(pointIds[i][j-1], pointIds[i][j])

        #sphereId = rs.AddSphere( (x * xScale, y * yScale, z * zScale), rSphere )
        #print(sphereId)
        #rs.ObjectColor(sphereId, [int(random.random() * 255), int(random.random() * 255),
int(random.random() * 255) ])

        j = j + 1
    i = i + 1

 # done!
```



Class01.10 along curve.py

```
rs.GetObject(“prompt”)
rs.GetInteger(“prompt”)
...
```

```
#! python3

import rhinoscriptsyntax as rs # this is the rhinoscript library
import scriptcontext as sc # configuration?

import System # standard base python
import System.Collections.Generic #collections specific library
import Rhino # Rhino

# Array points along a curve


crv_id = rs.GetObject("Select path curve")
if not crv_id: exit

count = rs.GetInteger("Number of items", 2, 2)
if not count: exit

points = rs.DivideCurve(crv_id, count)
for point in points: rs.AddPoint(point)
```

Class Assignment for next week

1. Do something innovative with a 1D, 2D, or 3D grid
    - Transformations of the grid
    - Different objects, grid on the points, etc.
    - Colors
    - Downstream operations, etc.

1. Run through https://www.w3schools.com/python up to arrays