Project explanation:

There is directory named pydir, that has two subdirectories 'x2+y2=z2' and 'x2+y2=z2-float'.

'x2+y2=z2' subdirectory contain files related to plot of XYZ points which are integer numbers as result of the x2+y2=z2 equation.

'x2+y2=z2-float' subdirectory contain files related to plot of XYZ points which are float numbers as result of the x2+y2=z2 equation.

```
root@ikali:~/pydir# ls
 BackupFiles  'x2+y2=z2'  'x2+y2=z2-float'
root@ikali:~/pydir# 
```

- **py1.py** file is Python program that shows results from x2+y2=z2 equation

- **BashPy** file is Bash Script (can be executed using command: ~/bash BashPy)

- **3D_Template.py** is file that has template Python code, but we need to add elements of lists x, y and z

- **3D_Plot_Points.py** is file as result of adding data inside **3D_Template.py** file. Then this file is ready to be executed and plot the results.

```
root@ikali:~/pydir# ls
 BackupFiles  'x2+y2=z2'  'x2+y2=z2-float'
root@ikali:~/pydir#
root@ikali:~/pydir#
root@ikali:~/pydir# cd 'x2+y2=z2'
root@ikali:~/pydir/x2+y2=z2# ls
3D_Plot_Points.py  BackupFiles  FutureUse  ResultingPlots
3D_Template.py     BashPy       py1.py
root@ikali:~/pydir/x2+y2=z2#
root@ikali:~/pydir/x2+y2=z2# 
```

Project execution:

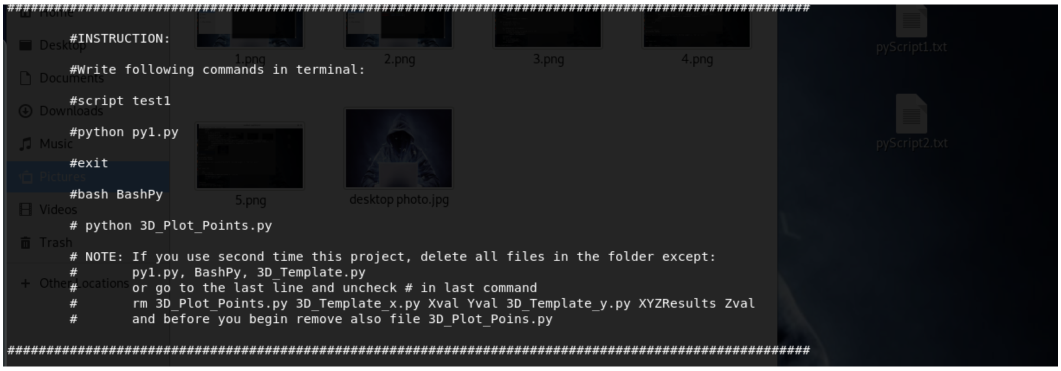
1. Using command ***script test1*** we will collect all Linux terminal output inside the file named ***test1***.

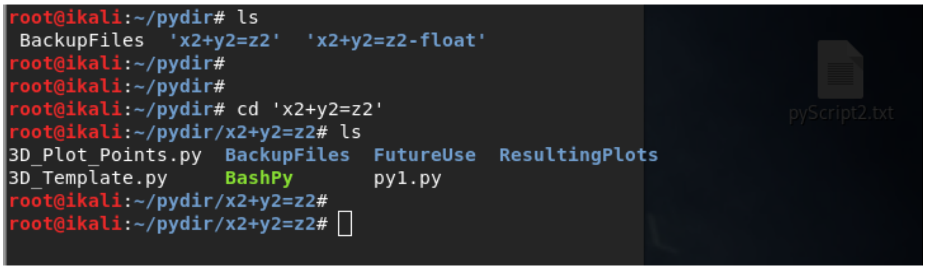
2. Now we will execute our first Python code from file ***py1.py*** using command ***python py1.py***.

3. When we use command ***exit*** we will stop execution of command ***script test1*** in Linux terminal and output of the executed ***py1.py*** Python code will be stored inside file ***test1*** - and no more recordings from the terminal window.

4. Next is to execute our Bash Script named ***BashPy*** using ***bash BashPy*** command. This script creates file named ***3D_Plot_Points.py***.

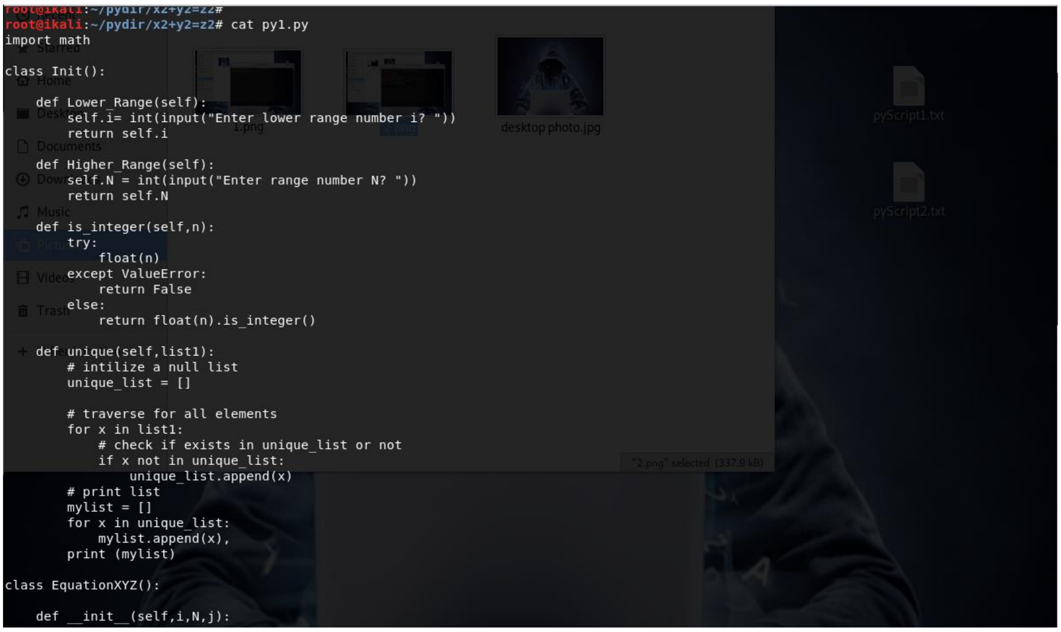
5. We execute our second Python code from ***3D_Plot_Points.py*** file using ***python 3D_Plot_Points.py*** command. Result is plot in 3D space.



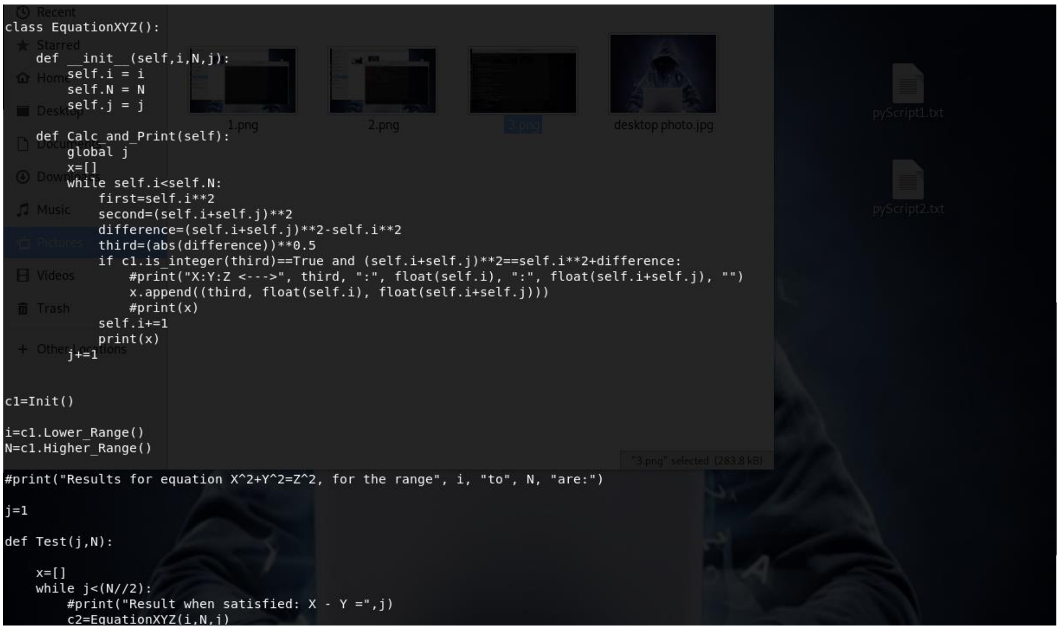


I. x2+y2=z2 equation solved with integer numbers

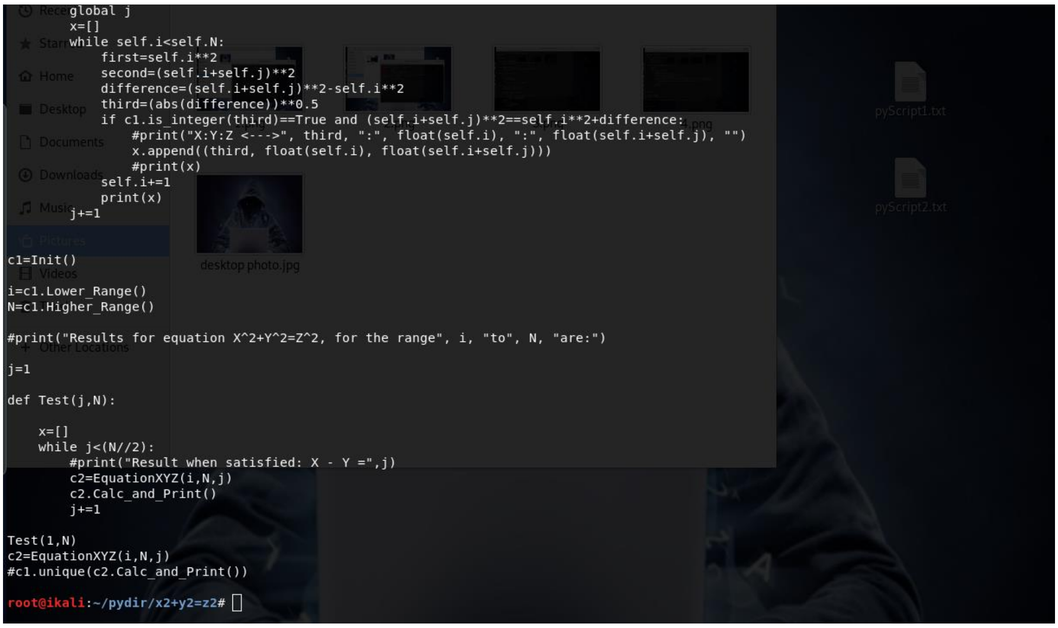
I.1. Content of our Python code, that solves x2+y2=z2 equation with results integer numbers, is inside of the py1.py file:



```
root@ikali:~/pydir/x2+y2=z2#
root@ikali:~/pydir/x2+y2=z2# cat py1.py
import math

class Init():

    def Lower_Range(self):
        self.i= int(input("Enter lower range number i? "))
        return self.i

    def Higher_Range(self):
        self.N = int(input("Enter range number N? "))
        return self.N

    def is_integer(self,n):
        try:
            float(n)
        except ValueError:
            return False
        else:
            return float(n).is_integer()

    def unique(self,list1):
        # intilize a null list
        unique_list = []

        # traverse for all elements
        for x in list1:
            # check if exists in unique_list or not
            if x not in unique_list:
                unique_list.append(x)
        # print list
        mylist = []
        for x in unique_list:
            mylist.append(x),
        print (mylist)

class EquationXYZ():

    def __init__(self,i,N,j):
```

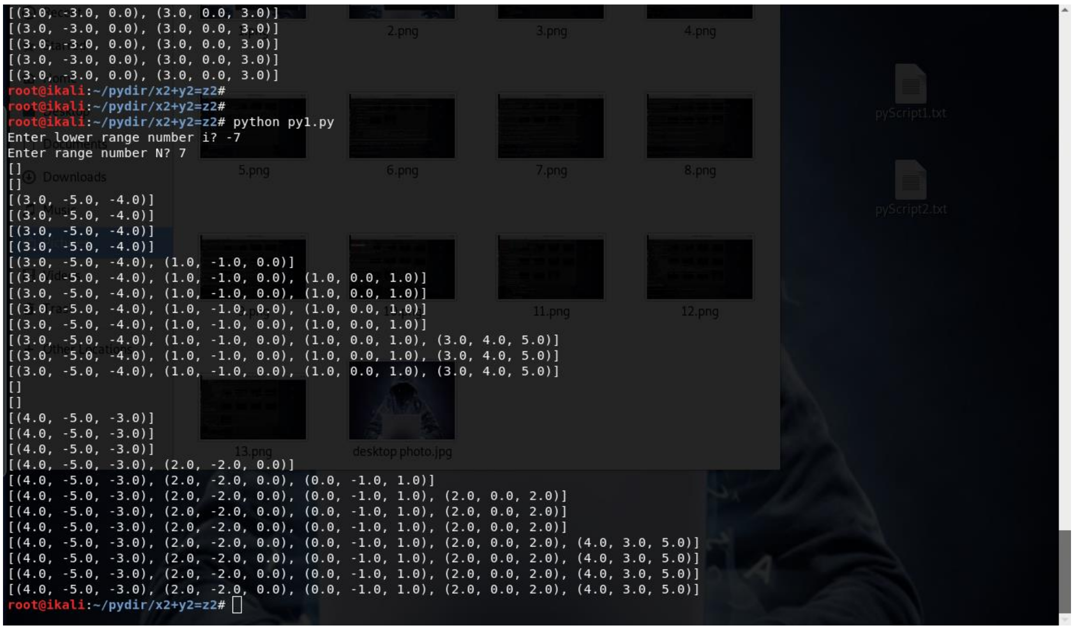


```
class EquationXYZ():

    def __init__(self,i,N,j):
        self.i = i
        self.N = N
        self.j = j

    def Calc_and_Print(self):
        global j
        x=[]
        while self.i<self.N:
            first=self.i**2
            second=(self.i+self.j)**2
            difference=(self.i+self.j)**2-self.i**2
            third=(abs(difference))**0.5
            if c1.is_integer(third)==True and (self.i+self.j)**2==self.i**2+difference:
                #print("X:Y:Z <--->", third, ":", float(self.i), ":", float(self.i+self.j), "")
                x.append((third, float(self.i), float(self.i+self.j)))
                #print(x)
            self.i+=1
            print(x)
        j+=1


c1=Init()

i=c1.Lower_Range()
N=c1.Higher_Range()

#print("Results for equation X^2+Y^2=Z^2, for the range", i, "to", N, "are:")

j=1

def Test(j,N):

    x=[]
    while j<(N//2):
        #print("Result when satisfied: X - Y =",j)
        c2=EquationXYZ(i,N,j)
```



```
        global j
        x=[]
        while self.i<self.N:
            first=self.i**2
            second=(self.i+self.j)**2
            difference=(self.i+self.j)**2-self.i**2
            third=(abs(difference))**0.5
            if c1.is_integer(third)==True and (self.i+self.j)**2==self.i**2+difference:
                #print("X:Y:Z <--->", third, ":", float(self.i), ":", float(self.i+self.j), "")
                x.append((third, float(self.i), float(self.i+self.j)))
                #print(x)
            self.i+=1
            print(x)
        j+=1

c1=Init()

i=c1.Lower_Range()
N=c1.Higher_Range()

#print("Results for equation X^2+Y^2=Z^2, for the range", i, "to", N, "are:")

j=1

def Test(j,N):

    x=[]
    while j<(N//2):
        #print("Result when satisfied: X - Y =",j)
        c2=EquationXYZ(i,N,j)
        c2.Calc_and_Print()
        j+=1

Test(1,N)
c2=EquationXYZ(i,N,j)
#c1.unique(c2.Calc_and_Print())

root@ikali:~/pydir/x2+y2=z2#
```

I.1.1 Example of the form of the output when we execute py1.py file, is shown bellow:



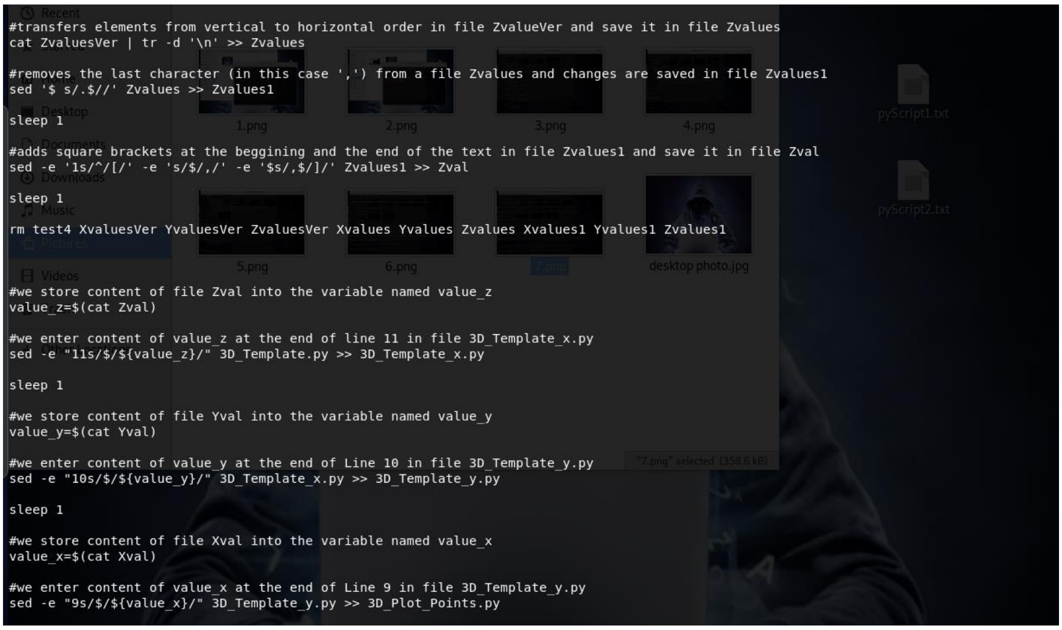
```
[(3.0, -3.0, 0.0), (3.0, 0.0, 3.0)]
[(3.0, -3.0, 0.0), (3.0, 0.0, 3.0)]
[(3.0, -3.0, 0.0), (3.0, 0.0, 3.0)]
[(3.0, -3.0, 0.0), (3.0, 0.0, 3.0)]
[(3.0, -3.0, 0.0), (3.0, 0.0, 3.0)]
root@ikali:~/pydir/x2+y2=z2#
root@ikali:~/pydir/x2+y2=z2#
root@ikali:~/pydir/x2+y2=z2# python py1.py
Enter lower range number i? -7
Enter range number N? 7
[]
[]
[(3.0, -5.0, -4.0)]
[(3.0, -5.0, -4.0)]
[(3.0, -5.0, -4.0)]
[(3.0, -5.0, -4.0)]
[(3.0, -5.0, -4.0), (1.0, -1.0, 0.0)]
[(3.0, -5.0, -4.0), (1.0, -1.0, 0.0), (1.0, 0.0, 1.0)]
[(3.0, -5.0, -4.0), (1.0, -1.0, 0.0), (1.0, 0.0, 1.0)]
[(3.0, -5.0, -4.0), (1.0, -1.0, 0.0), (1.0, 0.0, 1.0)]
[(3.0, -5.0, -4.0), (1.0, -1.0, 0.0), (1.0, 0.0, 1.0)]
[(3.0, -5.0, -4.0), (1.0, -1.0, 0.0), (1.0, 0.0, 1.0), (3.0, 4.0, 5.0)]
[(3.0, -5.0, -4.0), (1.0, -1.0, 0.0), (1.0, 0.0, 1.0), (3.0, 4.0, 5.0)]
[(3.0, -5.0, -4.0), (1.0, -1.0, 0.0), (1.0, 0.0, 1.0), (3.0, 4.0, 5.0)]
[]
[]
[(4.0, -5.0, -3.0)]
[(4.0, -5.0, -3.0)]
[(4.0, -5.0, -3.0)]
[(4.0, -5.0, -3.0), (2.0, -2.0, 0.0)]
[(4.0, -5.0, -3.0), (2.0, -2.0, 0.0), (0.0, -1.0, 1.0)]
[(4.0, -5.0, -3.0), (2.0, -2.0, 0.0), (0.0, -1.0, 1.0), (2.0, 0.0, 2.0)]
[(4.0, -5.0, -3.0), (2.0, -2.0, 0.0), (0.0, -1.0, 1.0), (2.0, 0.0, 2.0)]
[(4.0, -5.0, -3.0), (2.0, -2.0, 0.0), (0.0, -1.0, 1.0), (2.0, 0.0, 2.0)]
[(4.0, -5.0, -3.0), (2.0, -2.0, 0.0), (0.0, -1.0, 1.0), (2.0, 0.0, 2.0), (4.0, 3.0, 5.0)]
[(4.0, -5.0, -3.0), (2.0, -2.0, 0.0), (0.0, -1.0, 1.0), (2.0, 0.0, 2.0), (4.0, 3.0, 5.0)]
[(4.0, -5.0, -3.0), (2.0, -2.0, 0.0), (0.0, -1.0, 1.0), (2.0, 0.0, 2.0), (4.0, 3.0, 5.0)]
[(4.0, -5.0, -3.0), (2.0, -2.0, 0.0), (0.0, -1.0, 1.0), (2.0, 0.0, 2.0), (4.0, 3.0, 5.0)]
root@ikali:~/pydir/x2+y2=z2#
```

Because this output is hard to manage and to integrate into Python known data formats, we choose to save it as it is in a ***test1*** temporary file, and later we will process and arrange data as we need.

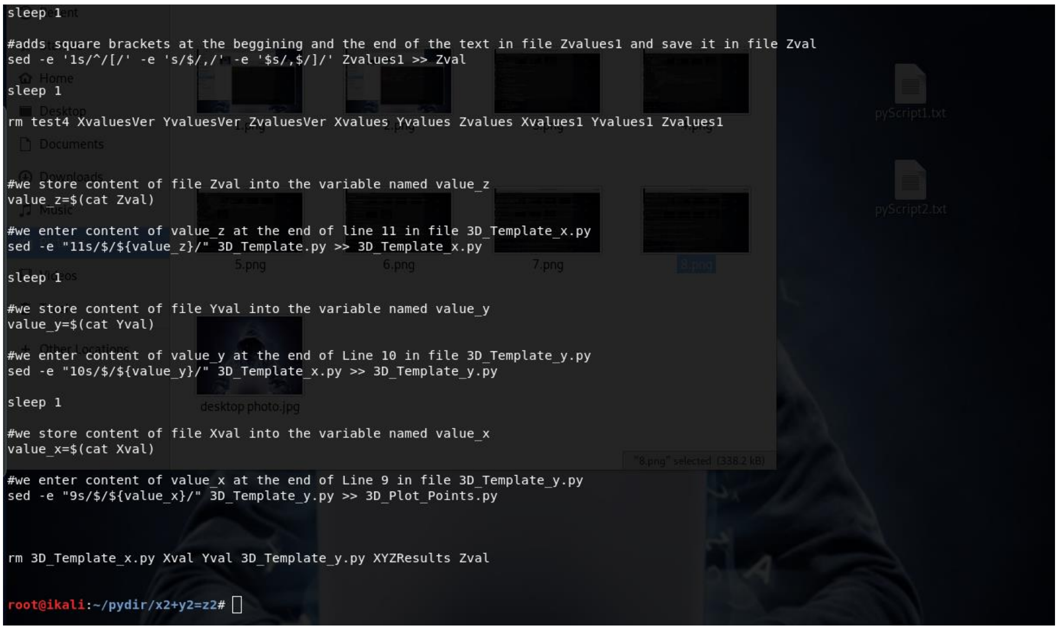
I.2. Content of Bash Script named ***BashPy*** is:

```
root@ikali:~/pydir/x2+y2=z2#
root@ikali:~/pydir/x2+y2=z2# cat BashPy
#!/bin/bash

#######################################################################################################

        #INSTRUCTION:

        #Write following commands in terminal:

        #script test1

        #python py1.py

        #exit

        #bash BashPy

        # python 3D_Plot_Points.py

        # NOTE: If you use second time this project, delete all files in the folder except:
        #       py1.py, BashPy, 3D_Template.py
        #       or go to the last line and uncheck # in last command
        #       rm 3D_Plot_Points.py 3D_Template_x.py Xval Yval 3D_Template_y.py XYZResults Zval
        #       and before you begin remove also file 3D_Plot_Poins.py

#######################################################################################################

#copies all text inside the parentesses () from file 'test1' and save it o the 'test2' file
sed -E -n 's/.*\((.*)\).*$/\1/p' test1 >> test2

#removes all duplicate lines in a file 'test2' and save unique lines in to file 'test2'
awk '!seen[$0]++' test2 >> test3

#removes all lines that end with 0.0
sed "/\0.0$/d" test3 >> XYZResults

sleep 2

rm test1 test2 test3
```

```
sleep 2

rm test1 test2 test3

#get first word in any line in file XYZResults and save it in file Xvalues
awk '{print $1}' XYZResults >> XvaluesVer

#transfers elements from vertical to horizontal order in file XvaluesVer and save it in file Xvalues
cat XvaluesVer | tr -d '\n' >> Xvalues

#removes the last character (in this case ',') from a file Xvalues and changes are saved in file Xvalues1
sed '$ s/.$//' Xvalues >> Xvalues1

sleep 1

#adds sqare brackets at the beggining and end of text in file Xvalues and save it in file Xval
sed -e '1s/^/[/' -e 's/$/,/' -e '$s/,$/]/' Xvalues1 >> Xval

#deletes first word in any line in file XYZResults and save it in file test4
awk '{ $1=""; print substr($0,2) }' XYZResults >> test4

#get first word in any line in file test4 and save it in file YvaluesVer
awk '{print $1}' test4 >> YvaluesVer

#transfers elements from vertical into horizontal order in file YvaluesVer and save it in file Yvalues
cat YvaluesVer | tr -d '\n' >> Yvalues

#removes the last character (in this vase ',') from a file Yvalues and changes are saved in file Yvalues1
sed '$ s/.$//' Yvalues >> Yvalues1

sleep 1

#adds square brackets at the beggining and the end of text in file Yvalues1 and save it in file Yval
sed -e '1s/^/[/' -e 's/$/,/' -e '$s/,$/]/' Yvalues1 >> Yval

#deletes first word in any line in file test4 and save it in file ZvaluesVer
awk '{print $1}' test4 >> ZvaluesVer

#transfers elements from vertical to horizontal order in file ZvalueVer and save it in file Zvalues
cat ZvaluesVer | tr -d '\n' >> Zvalues
```



```
#transfers elements from vertical to horizontal order in file ZvalueVer and save it in file Zvalues
cat ZvaluesVer | tr -d '\n' >> Zvalues

#removes the last character (in this case ',') from a file Zvalues and changes are saved in file Zvalues1
sed '$ s/.$//' Zvalues >> Zvalues1

sleep 1

#adds square brackets at the beggining and the end of the text in file Zvalues1 and save it in file Zval
sed -e '1s/^/[/' -e 's/$/,/' -e '$s/,$/]/' Zvalues1 >> Zval

sleep 1

rm test4 XvaluesVer YvaluesVer ZvaluesVer Xvalues Yvalues Zvalues Xvalues1 Yvalues1 Zvalues1

#we store content of file Zval into the variable named value_z
value_z=$(cat Zval)

#we enter content of value_z at the end of line 11 in file 3D_Template_x.py
sed -e "11s/$/${value_z}/" 3D_Template.py >> 3D_Template_x.py

sleep 1

#we store content of file Yval into the variable named value_y
value_y=$(cat Yval)

#we enter content of value_y at the end of Line 10 in file 3D_Template_y.py
sed -e "10s/$/${value_y}/" 3D_Template_x.py >> 3D_Template_y.py

sleep 1

#we store content of file Xval into the variable named value_x
value_x=$(cat Xval)

#we enter content of value_x at the end of Line 9 in file 3D_Template_y.py
sed -e "9s/$/${value_x}/" 3D_Template_y.py >> 3D_Plot_Points.py
```



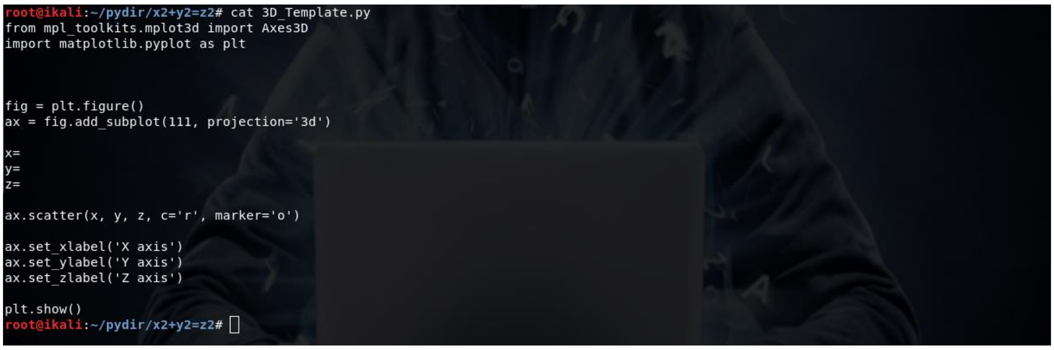
```
sleep 1

#adds square brackets at the beggining and the end of the text in file Zvalues1 and save it in file Zval
sed -e '1s/^/[/' -e 's/$/,/' -e '$s/,$/]/' Zvalues1 >> Zval

sleep 1

rm test4 XvaluesVer YvaluesVer ZvaluesVer Xvalues Yvalues Zvalues Xvalues1 Yvalues1 Zvalues1

#we store content of file Zval into the variable named value_z
value_z=$(cat Zval)

#we enter content of value_z at the end of line 11 in file 3D_Template_x.py
sed -e "11s/$/${value_z}/" 3D_Template.py >> 3D_Template_x.py

sleep 1

#we store content of file Yval into the variable named value_y
value_y=$(cat Yval)

#we enter content of value_y at the end of Line 10 in file 3D_Template_y.py
sed -e "10s/$/${value_y}/" 3D_Template_x.py >> 3D_Template_y.py

sleep 1

#we store content of file Xval into the variable named value_x
value_x=$(cat Xval)

#we enter content of value_x at the end of Line 9 in file 3D_Template_y.py
sed -e "9s/$/${value_x}/" 3D_Template_y.py >> 3D_Plot_Points.py


rm 3D_Template_x.py Xval Yval 3D_Template_y.py XYZResults Zval

root@ikali:~/pydir/x2+y2=z2#
```

How we manage data inside the temporary documents and how they are removed is shown in a lines with comments marked with #.

sleep commands are used temporary documents to be created and populated with data, before this temporary document is called with next line command. For exqmple ***sleep 1*** command delayes execution for 1 second of following command writen in the next line.

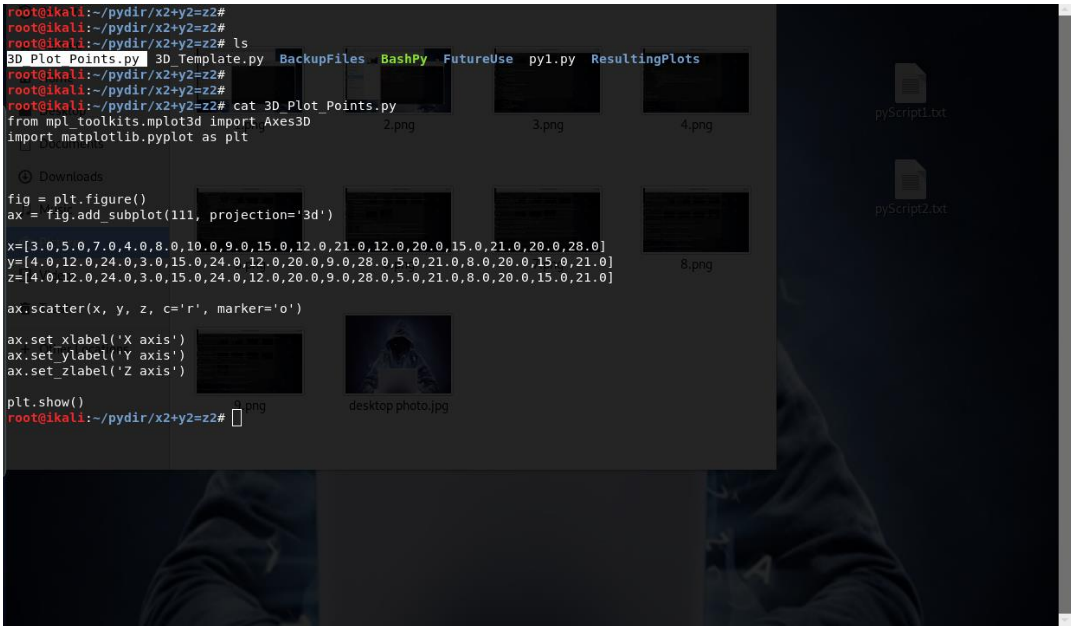
This Bash Script as result creates file ***3D_Plot_Points.py***, which later we execute.

I.3. Content of template file named ***3D_Template.py*** is:



After we execute ***BashPy*** file it processes the data, then it populates resulting data in a form of elements in x,y and z lists, inside ***3D_Plot_Points.py*** file.

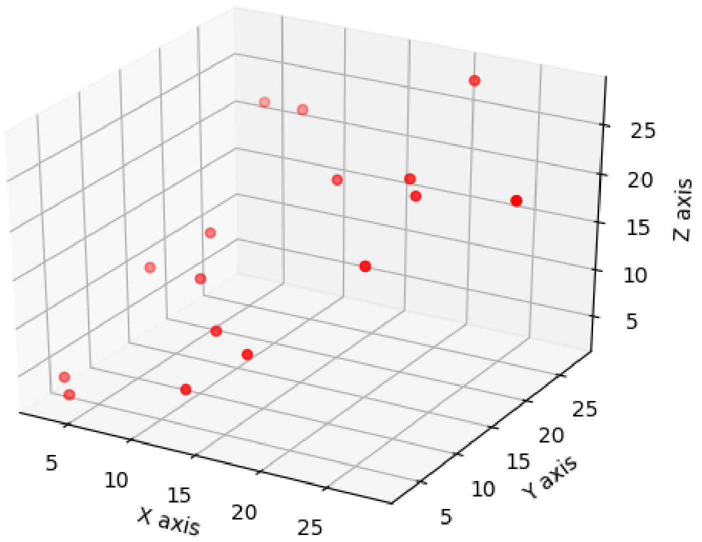
Content of our Python code inside the 3D_Plot_Points.py file, to plot XYZ dots is:



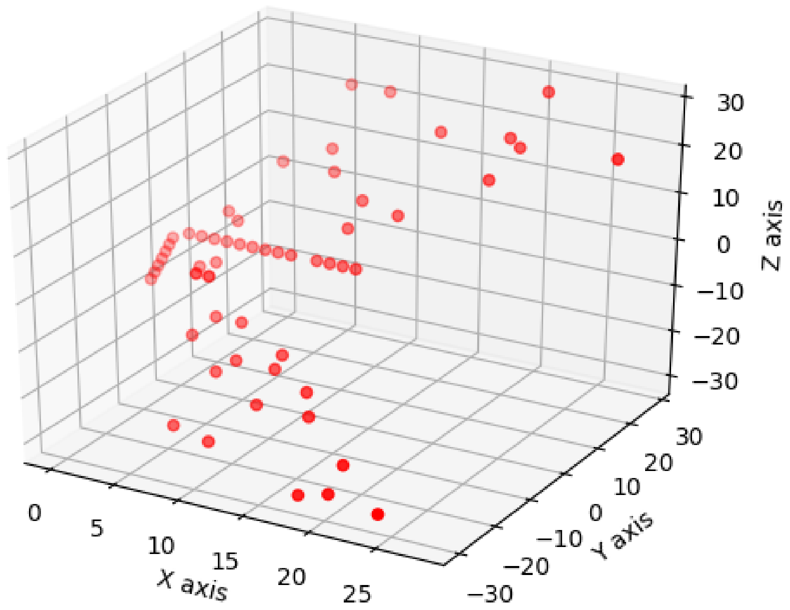
I.4 Plotting the resuts

If we follow instruction above for ***project execution***, as result we will receive one 3D plot with dots.

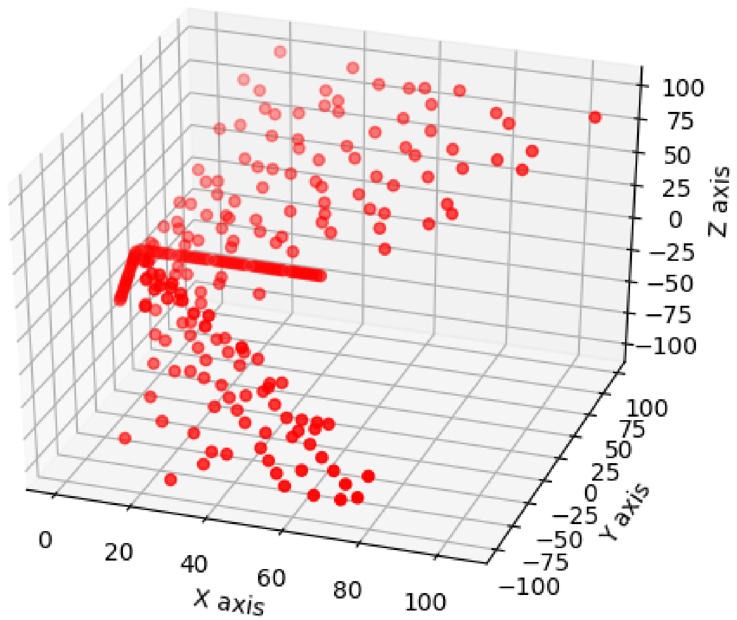
If we repeat overall procedure for different ranges of N, resulting plots are:



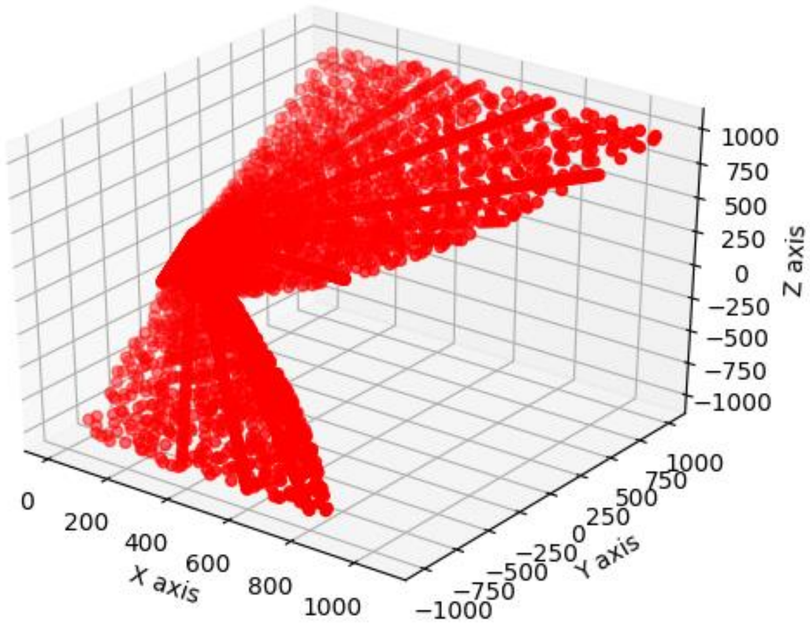
Plot 1 N number range is from 1 to 30



Plot 2 N number range is from -30 to 30



Plot 3 N number range is from -100 to 100

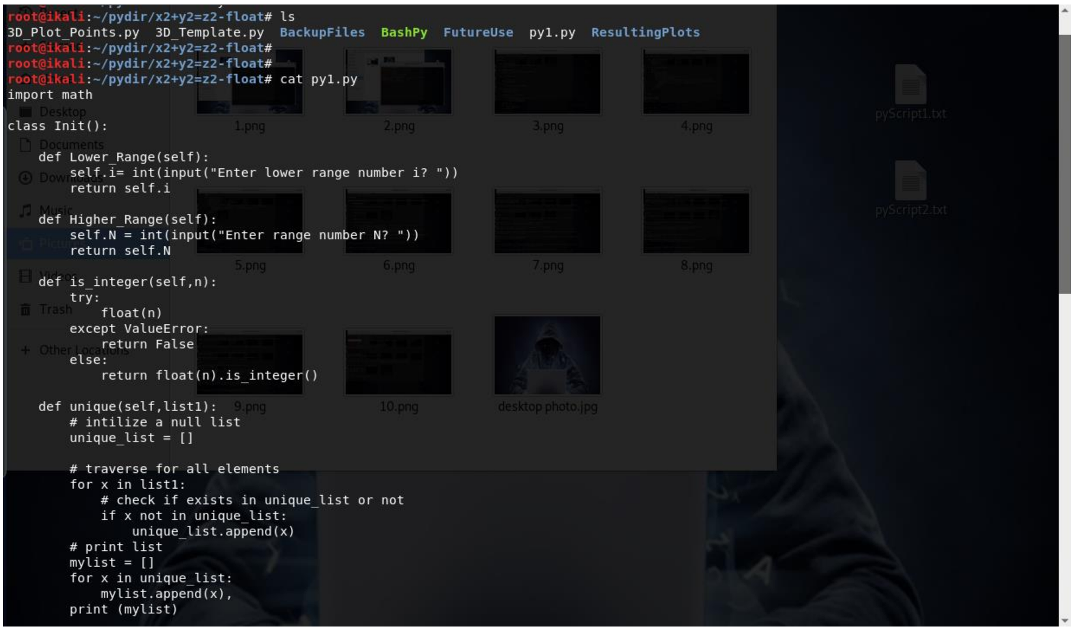


Plot 4 N number range is from -1000 to 1000

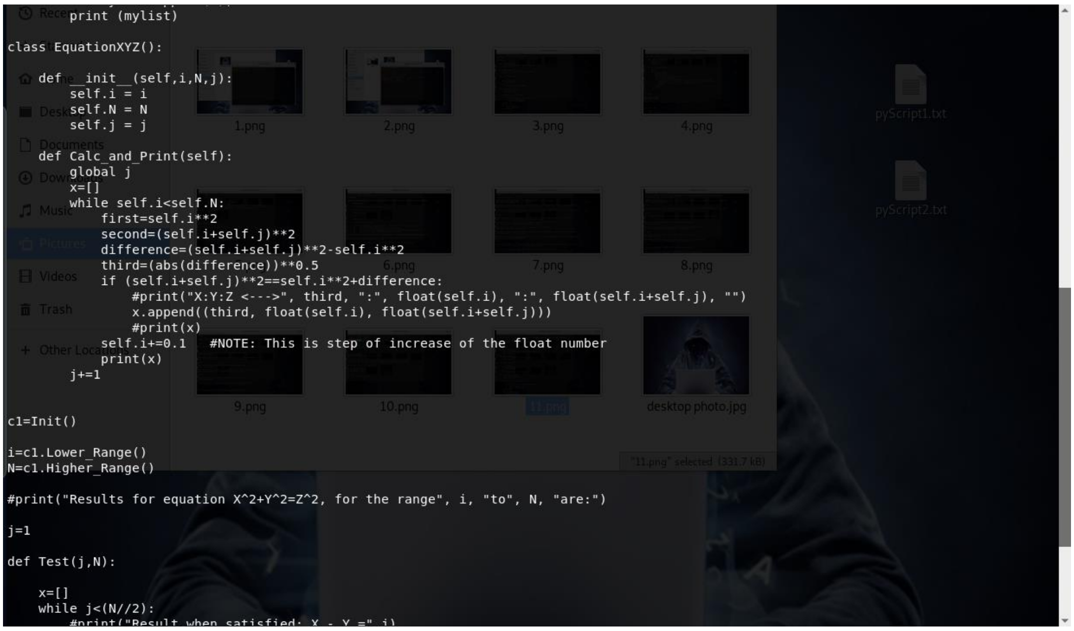
Resulting plots show us that all dots are in the same 2D plane.

II. x2+y2=z2 equation solved with float numbers

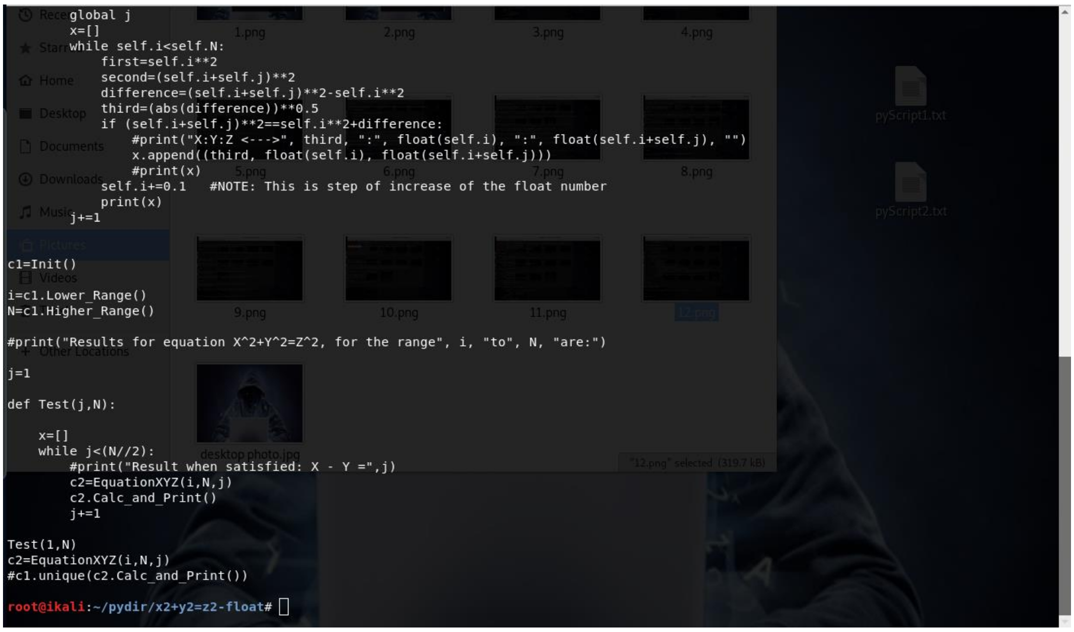
II.1 Changes are created inside py1.py file:



```
root@ikali:~/pydir/x2+y2=z2-float# ls
3D_Plot_Points.py  3D_Template.py  BackupFiles  BashPy  FutureUse  py1.py  ResultingPlots
root@ikali:~/pydir/x2+y2=z2-float#
root@ikali:~/pydir/x2+y2=z2-float#
root@ikali:~/pydir/x2+y2=z2-float# cat py1.py
import math

class Init():

    def Lower_Range(self):
        self.i= int(input("Enter lower range number i? "))
        return self.i

    def Higher_Range(self):
        self.N = int(input("Enter range number N? "))
        return self.N

    def is_integer(self,n):
        try:
            float(n)
        except ValueError:
            return False
        else:
            return float(n).is_integer()

    def unique(self,list1):
        # intilize a null list
        unique_list = []

        # traverse for all elements
        for x in list1:
            # check if exists in unique_list or not
            if x not in unique_list:
                unique_list.append(x)
        # print list
        mylist = []
        for x in unique_list:
            mylist.append(x),
        print (mylist)
```



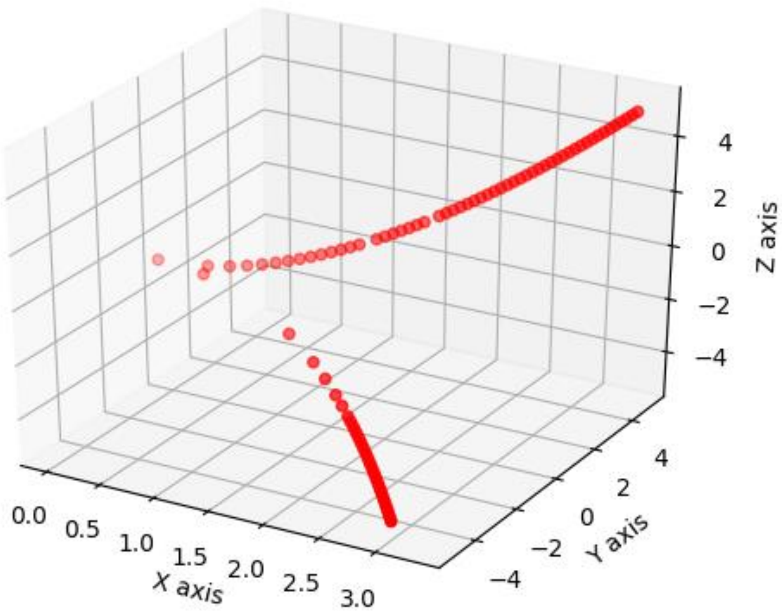
```
        print (mylist)

class EquationXYZ():

    def __init__(self,i,N,j):
        self.i = i
        self.N = N
        self.j = j

    def Calc_and_Print(self):
        global j
        x=[]
        while self.i<self.N:
            first=self.i**2
            second=(self.i+self.j)**2
            difference=(self.i+self.j)**2-self.i**2
            third=(abs(difference))**0.5
            if (self.i+self.j)**2==self.i**2+difference:
                #print("X:Y:Z <--->", third, ":", float(self.i), ":", float(self.i+self.j), "")
                x.append((third, float(self.i), float(self.i+self.j)))
                #print(x)
            self.i+=0.1   #NOTE: This is step of increase of the float number
            print(x)
        j+=1

c1=Init()

i=c1.Lower_Range()
N=c1.Higher_Range()

#print("Results for equation X^2+Y^2=Z^2, for the range", i, "to", N, "are:")

j=1

def Test(j,N):

    x=[]
    while j<(N//2):
```



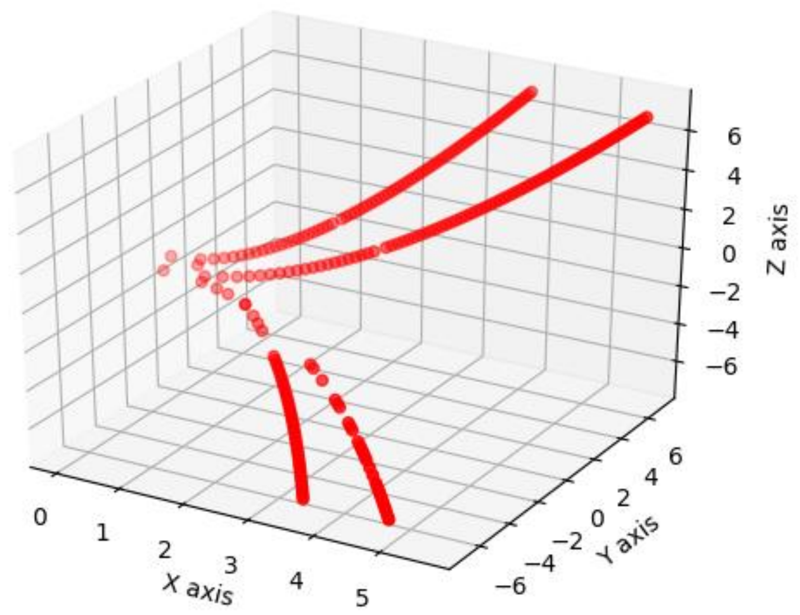
```
    global j
    x=[]
    while self.i<self.N:
        first=self.i**2
        second=(self.i+self.j)**2
        difference=(self.i+self.j)**2-self.i**2
        third=(abs(difference))**0.5
        if (self.i+self.j)**2==self.i**2+difference:
            #print("X:Y:Z <--->", third, ":", float(self.i), ":", float(self.i+self.j), "")
            x.append((third, float(self.i), float(self.i+self.j)))
            #print(x)
        self.i+=0.1   #NOTE: This is step of increase of the float number
        print(x)
    j+=1

c1=Init()
i=c1.Lower_Range()
N=c1.Higher_Range()

#print("Results for equation X^2+Y^2=Z^2, for the range", i, "to", N, "are:")

j=1

def Test(j,N):

    x=[]
    while j<(N//2):
        #print("Result when satisfied: X - Y =",j)
        c2=EquationXYZ(i,N,j)
        c2.Calc_and_Print()
        j+=1

Test(1,N)
c2=EquationXYZ(i,N,j)
#c1.unique(c2.Calc_and_Print())

root@ikali:~/pydir/x2+y2=z2-float#
```

II.2 Plotting the resuts

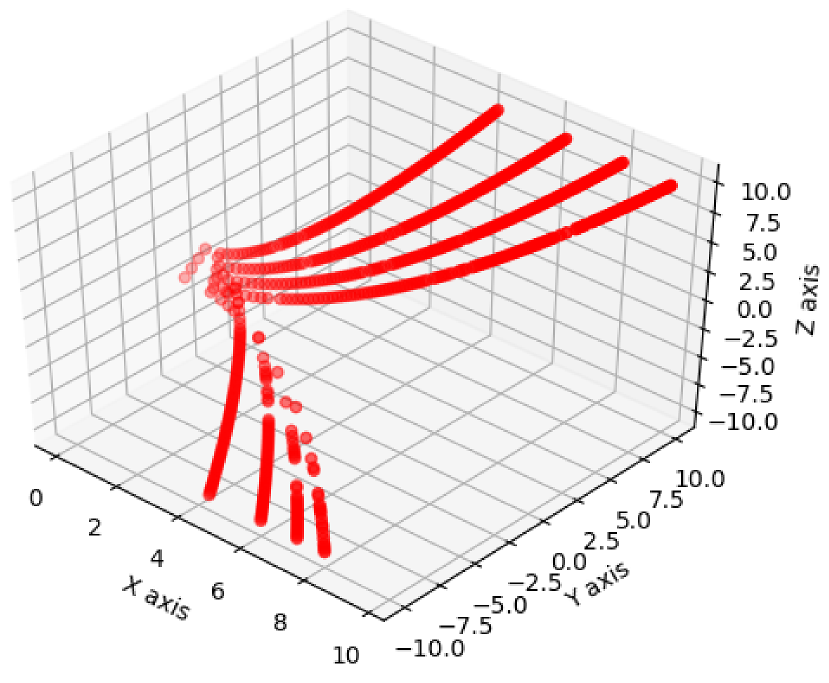
If we repeat overall procedure for different ranges of N, resulting plots are:



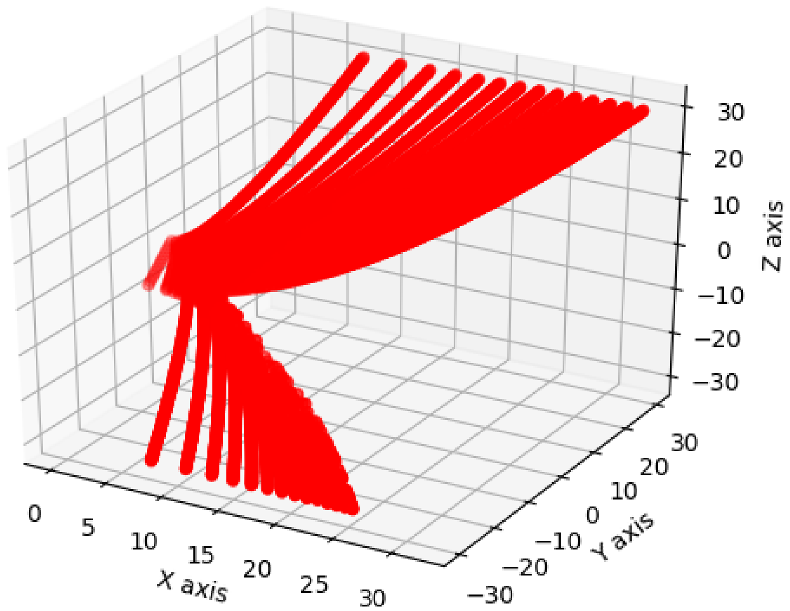
Plot 1 N number range is from -5 to 5, step to next number is 0.1



Plot 2 N number range is from -7 to 7, step to next number is 0.1



Plot 3 N number range is from -10 to 10, step to next number is 0.1



Plot 4 N number range is from -30 to 30, step to next number is 0.1

Resulting plots show us that all dots are in the same 2D plane.

III. In similar way we can find complex numbers that satisfy x2+y2=z2 equation

Plotting can be done as group of three dots: {Re(x), Im(x)}, {Re(y), Im(y)} and {Re(z), Im(z)} in complex plane. If we connect these three dots we will have line (if x or y is 0) or triangle (x,y,z != 0) as resulting object. Any other complex numbers as solution to the x2+y2=z2 equation will create group of dots that can be plotted as well as an object.