Actividad 2: Elementos de la programación Python 1

Martin Alejandro Paredes Sosa

Febrero 2016

1. Introducción

Esta practica consistio en la realización de los primeros programas en Python del curso de *Física Computacional*. Python es un lenguaje de programación interpretado cuya filosofia se enfoca en la sintaxis que favorezca el codigo legible. Es un lenguaje multiparadigma, ya que soporta lenguaje orientado a objetos, programación imperativa y en menor medida programación funcional. [1]

2. Problemas Realizados

Como ya se menciono antes, esta actividad consistio en la realización de los primeros programas en Python, lo cuales nos permitieron tener un mejor entendimiente de como funciona el lenguaje.

Se realizaron un total de 5 problemas, en los cuales se utilizaba diferentes herramientas de Python y sus librerias.

2.1. Problema 1

"Se deja caer una pelota desde el techo de una torre de altura h. Se desea saber la altura de la pelota respecto a la torre a un determinado tiempo después de haber sido dejada caer."[2]

Programa original

```
h = float(input("Proporciona la altura de la torre: "))
t = float(input("Ingresa el tiempo: "))
s = 0.5*9.81*t**2
print("La altura de la pelota es", h-s, "metros")
```

En base a este codigo, se modifico para que se introduzca la altura(m) de la torre y se obtenga un tiempo de vuelo.

Programa obtenido: caida.py

```
#Importar funcion sqtr
from math import sqrt

#Pedir al altura a usuario
h = float(input("Proporciona la altura de la torre: "))

g=9.81 #Constate de la gravedad

t=sqrt(2*h/g) #Calculo de tiempo de caida

#Impresion de Resultados
print'El tiempo de caida para una altura ',h,'m es: ', t
```

El resultado del programa fue

```
In [1]: run caida.py
Proporciona la altura en metros de la torre: 10
El tiempo de caida para una altura  10.0 m es:  1.42784312293 s
```

2.2. Problema 2

" Un satélite orbita la Tierra a una altura h, con un periodo T en segundos. Demuestre que la altitud h del satélite sobre la superficie de la Tierra esta dado por la expresión:

$$(R+h)^3 = \frac{GMT^2}{4\pi^2} \tag{1}$$

donde $G = 6.67 \times 10^{-11} \frac{Nm^2}{kg^2}$ es la constante de Gravitación Universal de Newton, $M = 5.97 \times 10^{24} kg$ es la masa de la Tierra y $R = 6371 km$ es su radio."[2]

Demostración Suponiendo una órbita circular, tendremos un satélite que orbite a velocidad constante, aplicando la Segunda Ley de Newton, considerando que la aceleración sería centrípeta:

$$F = m\frac{v^2}{r} \tag{2}$$

Siendo F la fuerza gravitatoria entre la Tierra y el satélite:

$$F = \frac{GMm}{r^2} \tag{3}$$

Expresamos a la velocidad en términos del período del satélite:

$$v = \frac{2\pi r}{T} \tag{4}$$

Sustituyendo v en (2) e igualando con (3) obtenemos:

$$\frac{GMm}{r^2} = m\frac{\left(\frac{2\pi r}{T}\right)^2}{r} \tag{5}$$

En el caso más general de un satélite orbitando un planeta, la altura de su órbita no es despreciable respecto al radio de la órbita, por lo que la distancia r es igual a la suma del radio del planeta R más la altura h de la órbita sobre su superficie. Por lo que $r = (R+h)$. Simplificando (5) y sustituyendo a r, llegamos a (1):

$$(R+h)^3 = \frac{GMT^2}{4\pi^2} \tag{6}$$

Para este problema se creo un programa con el cual podremos identicar la altura h de un setelite con un periodo T definido por el usuario.

Programa creado: Satelite.py

```
#Impotar el valor de pi
from math import pi

#Definicion de constantes
G = 6.67*(10**(-11)) #Constante de gravitacion
M = 5.97*(10**(24)) #Masa de la tierra en kg
R = 6371000  #Radio de la tierra en m

#Pedir el perido al usuario
t = float(input('Proporciona el periodo del satelite (minutos)'))
#Tranformar el periodo de min -> s
T = t*60

#Calculo de la altura desde la superficie de la tierra de la orbita
H = ((G*M*T**2)/(4*pi**2))**(1./3.)-R

print 'La altura a la que orbita un satelite de periodo ',
 t,'minutos es de ', H,'metros'
```

Los resultados obtenidos:

24 Horas

```
In [1]: run Satelite.py
Proporciona el periodo del satelite (minutos)1440
La altura a la que orbita un satelite de periodo  1440.0 minutos es de  35855910.1
```

90 minutos

```
In [2]: run Satelite.py
Proporciona el periodo del satelite (minutos)90
La altura a la que orbita un satelite de periodo  90.0 minutos es de  279321.62537
```

45 minutos

```
In [3]: run Satelite.py
Proporciona el periodo del satelite (minutos)45
La altura a la que orbita un satelite de periodo  45.0 minutos es de
 -2181559.89781 metros
```

Este ultimo resultado nos indica que no es posible alcanzar este periodo en ninguna altura fuera de la tierra.

2.3. Problema 3

"Un punto en el espacio en el sistema de coordenadas polares se describe por las cantidades (r, θ). La relación entre coordenadas polares y el sistema de coordenadas cartesianos, esta dada por las ecuaciones: $x = rcos\theta, y = rsin\theta$. Ingrese el siguiente programa para calcular las coordenadas cartesianas a partir de las coordenadas polares."[2]

Programa original

```
from math import sin,cos,pi
r = float(input("Introduce r: "))
d = float(input("Ingresa theta en grados: "))
theta = d*pi/180
x = r*cos(theta)
y = r*sin(theta)
print("x =",x," y =",y)
-----------
Introduce r: 5
Ingresa theta en grados: 30
x = 4.330127018922194  y = 2.4999999999999996
```

Se creo un programa que calculara las coordenadas esfericas (r, θ, ϕ), a partir de las coordenas cartesianas (x, y, z).

Programa creado: CordEsferica.py [3]

```
#Importar Funciones necesarias
from math import acos, atan, sqrt

x = float(input('Componente x: '))
y = float(input('Componente y: '))
z = float(input('Componente z: '))

#Calculo de las coordenadas
r = sqrt(x*x + y*y + z*z) #Norma
   #Angulos
theta = acos(z/r)
phi = atan(y/x)

print 'r= ',r
print 'theta= ', theta
print 'phi=  ', phi
```

Resultados:

```
In [1]: run CordEsferica.py
Componente x: 3
Componente y: 4
Componente z: 2
r=  5.38516480713
theta=  1.19028994968
phi=   0.927295218002
```

2.4. Problema 4

"Números pares (even) e impares(odd). Sabemos que los números pares son divisibles entre 2, es decir que su residuo es cero $(2n)$, mientras que los impares tienen residuo 1 $(2n + 1)$. Nos apoyamos en la operación de módulo %." [2]

```
n = int(input("Enter an integer: "))
if n%2==0:
     print("even")
```

```
else:
     print("odd")
-----------
Enter an integer: 6
even
```

"Ingresa el siguiente código que admite 2 enteros, que se utiliza el control WHILE (mientas que). Comprende su dinámica."

Programa creado en base al original: ParImpar.py

```
print 'Ingrese dos numeros enteros, un par y un impar'

m = int(input('Ingrese primer numero entero: '))
n = int(input('Ingrese segundo numero entero: '))

while (m+n)%2==0:
    print 'Uno debe ser impar y el otro par, vuelva a intentar'
    m = int(input('Ingrese primer numero entero: '))
    n = int(input('Ingrese segundo numero entero: '))

print 'Los numeros escogidos son ',m,'y ',n
```

Resultados:

```
In [1]: run ParImpar.py
Ingrese dos numeros enteros, un par y un impar
Ingrese primer numero entero: 16
Ingrese segundo numero entero: 2
Uno debe ser impar y el otro par, vuelva a intentar
Ingrese primer numero entero: 15
Ingrese segundo numero entero: 3
Uno debe ser impar y el otro par, vuelva a intentar
Ingrese primer numero entero: 10
Ingrese segundo numero entero: 5
Los numeros escogidos son  10 y  5
```

2.5. Problema 5

"Los Números de Fibonacci es una sucesión de números enteros aparecen en toda la naturaleza.

El siguiente programa calcula la secuencia de Fibonacci, introduce la condición de control while (mientras que)"

Programa original

```
f1,f2 = 1,1
while f2<1000:
      print(f2)
      f1,f2 = f2,f1+f2
```

Basándonos en la idea de el programa de la serie se Fibonacci, se creó uno para la secuencia de números de Catalan, dados por la fórmula de recurrencia:

$$C_0 = 1, C_{(n+1)} = \frac{2(2n+1)}{(n+2)} C_n$$

Programa creado: Catalan.py

```
x1,  n =  1 , 0

while (x1 < 1E+6):
    print x1
    x1 = x1*(4*n+2)/(n+2)
    n+=1
```

Resulatado:

```
In [1]: run Catalan.py
1
1
2
5
14
42
132
429
1430
4862
16796
58786
208012
742900
```

Referencias

[1] Wikipedia,(2016) *Python*. Recuperado de https://es.wikipedia.org/wiki/Python

[2] Lizárraga, C. (2016) *Actividad 2 (2016-1)*. Recuperado de http://computacional1.pbworks.com/w/page/104476954/Actividad%202%20(2016-1)

[3] Wikipedia,(2016) *Spherical coordinate system*. Recuperado de https://en.wikipedia.org/wiki/Sphericalcoordinatesystem#Coordinatesystemconversions