Actividad 2: Elementos de la programación Python 1

Ana Gabriela Carretas Talamante

27 de enero de 2016

1. Introducción

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible [1]. En esta actividad se procuraron códigos en lenguaje Python con la misión de mostrar el uso de las variables y la redacción en el lenguaje designado. Se presentarán entonces las modificaciones realizadas a cada ejemplo de programa, sugeridos en la página de la materia [2].

2. 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."

Programa original: caida.py

```
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")
-----------
Proporciona la altura de la torre: 10
Ingresa el tiempo: 1.42
('La altura de la pelota es', 0.10955799999999982, 'metros')
```

Modificando el código anterior, se obtuvo uno en el que tras pedir la altura h en metros, imprimiera el tiempo en segundos que tarda en llegar la pelota al suelo.

Programa modificado

```
from math import sqrt
h = float(input("Proporciona la altura de la torre: "))
t = sqrt(2*h/9.8)
print("La pelota tarda en caer", t, "segundos")
-----------
Proporciona la altura de la torre: 10
('La pelota tarda en caer', 1.4285714285714286, 'segundos')
```

3. 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 $G = 6.67 \times 10^{-11} \frac{m^3}{kgs^2}$ es la constante de Gravitación Universal de Newton, $M = 5.97 \times 10^{24} kg$ es la masa de la Tierra y $R = 6371 km$ su radio."

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 [3]. Por lo que $r = (R + h)$. Simplificando (5) y sustituyendo a r, llegamos a (1):

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

Se escribió un programa donde imprimía la altura h en metros del satélite con órbita de período T, el cual se pide al usuario.

Programa creado

```
from math import pi
T = float(input("Proporciona el período de órbita en segundos: "))
k = (6.67e-11*5.97e24)/(4*pi*pi)
h = (k*T*T)**(1./3.)-6371000
print("La altura del objeto con respecto a la superficie terrestre es",
h, "metros")
```

Este arrojó resultados para los tiempos:

- **24 horas:**

  ```
  Proporciona el período de órbita en segundos: 86400
  ('La altura del objeto con respecto a la superficie terrestre es',
  35855910.176174976, 'metros')
  ```

- **90 minutos:**

  ```
  Proporciona el período de órbita en segundos: 5400
  ('La altura del objeto con respecto a la superficie terrestre es',
  279321.6253728606, 'metros')
  ```

- **45 minutos:**

  ```
  Proporciona el período de órbita en segundos: 2700
  ('La altura del objeto con respecto a la superficie terrestre es',
  -2181559.8978108233, 'metros')
  ```

4. 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. "

Programa original: Polar.py

```
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 creó un programa que a partir de introducir coordenadas rectangulares (x, y, z), imprimiera esféricas (ρ, θ, ϕ).

Programa creado

```
from math import atan,acos,pi,sqrt
x = float(input("Introduce x: "))
y = float(input("Introduce y: "))
z = float(input("Introduce z: "))
rho = sqrt((x*x)+(y*y)+(z*z))
theta = atan(y/x)
phi = acos(z/(sqrt((x*x)+(y*y)+(z*z))))
print("rho =",rho," theta =",theta," phi =",phi)
-----------
Introduce x: 4
Introduce y: 6
Introduce z: 3
rho = 7.810249675906654  theta = 0.982793723247329  phi = 1.176551951730111
```

5. 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 %."

```
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 original: EvenOdd.py

```
print("Enter two integers, one even, one odd.")
m = int(input("Enter the first integer: "))
n = int(input("Enter the second integer: "))
while (m+n)%2==0:
    print("One must be even and the other odd.")
    m = int(input("Enter the first integer: "))
    n = int(input("Enter the second integer: "))
print("The numbers you chose are",m,"and",n)
-----------
Enter two integers, one even, one odd.
Enter the first integer: 5
Enter the second integer: 5
One must be even and the other odd.
Enter the first integer: 5
Enter the second integer: 6
The numbers you chose are 5 and 6
```

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: Fibonacci.py

```
f1,f2 = 1,1
while f2<1000:
      print(f2)
      f1,f2 = f2,f1+f2
-----------
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
```

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

```
n,c = 0,1
while c<100000:
      print(c)
      n,c = n+1,(((4*n)+2)/(n+2))*c
-----------
1
1.0
2.0
5.0
14.0
42.0
132.0
429.0
1430.0
4862.0
16796.0
58786.0
```

Referencias

[1] Wikipedia, *Python*. Recuperado el 27 de enero de 2016 de https://es.wikipedia.org/wiki/Python

[2] Lizárraga, C. *Actividad 2 (2016-1)*. Recuperado el 27 de enero de 2016 de http://computacional1.pbworks.com/w/page/104476954/Actividad%202%20(2016-1)

[3] *Gravitación y tercera Ley de Kepler*. Recuperado el 27 de enero de 2016 de http://educativa.catedu.es/44700165/aula/archivos/repositorio/3000/3227/html/22_gravitacin_y_tercera_ley_de_kepler.html