Universidad del Estado de Sonora

División de Ciencias Exactas y Naturales.

Licenciatura en Física.

Física Computacional 1

Elementos de la programación Python 1

Olga María Fimbres Morales

28 de Enero de 2016

Introducción.

Anteriormente ya hemos trabajado con lenguajes de programación, como lo es Fortran, los cuales nos han ayudado en la realización de diferentes programas sencillos que nos permiten realizar una operación matemática de una forma sencilla además de tener la posibilidad de realizar una representación de dicho problema. Siendo entonces una herramienta sumamente útil para una gran gama de ciencias.
De igual modo, aprender un nuevo lenguaje de programación, al igual que lo es aprender un nuevo idioma, nos da la posibilidad de resolver aun más problemas pues ahora contamos con un mayor número de herramientas para abórdalos, ampliando así nuestras opciones para trabajar.

Python

Python[1] es un lenguaje de programación interpretado creado en 1991 por Guido van Rossum, es multiparadigma pues soporta orientación a objetos, programación imperativa y programación funcional.
Existe una filosofía Python, la cual enfoca al lenguaje a ser simple, explicito, legible y práctico; lo que hace que sea sumamente sencillo de manejar incluso para principiantes pues facilita su manejo al utilizar palabras donde otros lenguajes utilizan símbolos.
Posee además un modo interactivo en el que es posible ingresar los comandos uno a uno e ir observando sus resultados, lo que permite probar partes de un código antes de implementarlo por completo.
Finalmente, cabe señalar que Python cuenta con una amplia biblioteca estándar las cuales incluyen un gran número de herramientas que son capaces de interactuar con otros lenguajes. Todo esto, y más, hace a Python una gran herramienta utilizable para casi cualquier persona.

Problema 1

En el primer problema se nos proporciona un sencillo programa utilizando Python el cual nos permite conocer la posición, a partir del techo de un edificio, partiendo de conocer la altura h de dicho edificio y el tiempo t para el cual deseamos conocer la posición:

```
Programa: 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")
```

A partir de este código es posible modificarlo para conocer ahora el tiempo en que una pelota llegara al suelo a partir de proporcionar la altura del edificio realizando un simple despeje de la ecuación anterior para $s = h$; siendo importante señalar que es necesario importar la función raíz cuadrada desde una de sus bibliotecas:

```
from math import sqrt
print("Problema 1")
h = float(input("Proporciona la altura de la torre en metros: "))
t = sqrt(2*h/9.81)
print("El tiempo en que la pelota llega al suelo son",t,"segundos")
```

Imprimiendo como resultado lo siguiente:

```
Problema 1
Proporciona la altura de la torre en metros: 50
('El tiempo en que la pelota llega al suelo son', 3.19, 'segundos')
```

Problema 2

En este problema se nos pide hacer un pequeño código capaz de arrojar como resultado la altura h que debería de tener un satélite a partir de la superficie terrestre para realizar un periodo alrededor de la Tierra en diferentes tiempos T, es cual se nos es pedido en minutos.
Notese que esta vez fue necesario importar el valor de pi para realizar las operaciones siendo suficiente especificar todas las demás constantes que se usarían.

```
from math import pi
print ("Problema 2")
T = 60*float(input("Proporciona el periodo de un satélite alrededor
de la Tierra en minutos: "))
G = 6.67e-11
M = 5.97e24
R = 6371000.
h = (((G*M*T**2)/(4*pi**2))**(1.0/3.0))- R
print ("Su altura debe ser de", h, "metros")
```

Una vez ejecutado el programa se nos presenta en pantalla que indiquemos el periodo deseado en minutos, ya que dado el funcionamiento del código este transforma los minutos especificados a segundos para poder llevar a cabo un cálculo correcto del resultado:

```
Problema 2
Proporciona el periodo de un satélite alrededor de la Tierra en minutos:
90
('Su altura debe ser de', 279321.6253728606, 'metros')
Problema 2
Proporciona el periodo de un satélite alrededor de la Tierra en minutos:
45
('Su altura debe ser de', -2181559.8978108233, 'metros')
Problema 2
Proporciona el periodo de un satélite alrededor de la Tierra en minutos:
24*60
('Su altura debe ser de', 35855910.176174976, 'metros')
```

Problema 3

A partir de un programa para calcular las coordenadas cartesianas a partir de las coordenadas polares se produjo un nuevo programa que a partir de las coordenadas cartesianas proporcionara las coordenadas esféricas[2] correspondientes.

```
Programa: 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)
```

Se siguió el mismo esquema que es programa anterior, solo que fue necesario importar diferentes identidades trigonométricas debido a las relaciones que existen entre las coordenadas cartesianas y las esféricas; en general, a partir de un programa base, resultó sumamente sencillo realizar este nuevo programa donde lo importante recae en no olvidar importar dichas funciones.

```
from math import sin,acos,pi, atan, sqrt
print ("Problema 3")
print ("Especifica las coordenadas cartesianas del punto: ")
x = float(input("Introduce x: "))
y = float(input("Introduce y: "))
z = float(input("Inytoduce z: "))
r = sqrt(x**2 + y**2 + z**2)
theta = acos(z/r)
phi = atan(y/x)
print ("Las coordenas esféricas correspondientes son:")
print("r =",r,"theta =",theta,"phi =",phi)
```

Un ejemplo de los resultados:

```
Problema 3
Especifica las coordenadas cartesianas del punto:
Introduce x: 1
Introduce y: 1
```

```
Inytoduce z: 2
Las coordenas esféricas correspondientes son:
('r =', 2.449489742783178, 'theta =', 0.6154797086703871, 'phi =', 0.7853981633974
```

Problema 4

Este problema solo nos pedía comprender la forma en que un programa funciona al tener estos condicionales para mostrar o no un resultado. Por ejemplo:

```
print ("Problema 4b"),
print("Ingrese dos números, uno par y uno impar:")
m = int(input("Ingrese el primer número: "))
n = int(input("Ingrese el segundo número: "))
while (m+n)%2==0:
    print("Uno debe ser par y el otro impar.")
    m = int(input("Ingrese el primer número: "))
    n = int(input("Ingrese el segundo número: "))
print("Los números que escogió son",m,"y",n)
```

este código posee una condicionante *while*, es decir que realizara una tarea siempre y cuando se siga cumpliendo la condición especificada. Por otro lado, el siguiente programa:

```
n = int(input("Ingrese un número: "))
if n%2==0:
     print("Par")
else:
     print("Impar")
```

posee una condición que le permite presentar dos resultados diferentes dependiendo de si la respuesta la condición es positiva o no.

Problema 5

Python también nos permite realizar programas donde se encuentren implicados los ciclos para poder realizar múltiples veces una misma operación o, como en este caso, mostrar una sucesión de números.
El siguiente programa nos permite obtener los números de la secuencia de Fibonacci que son menores a 1000, pero pueden obtenerse más o menos números como se desee, únicamente debe ser cambiada la condición que se aplica para que el programa calcule los resultados:

```
from math import factorial
print("Problema 5")
print("Secuencia de Fibonacci.")
f1,f2 = 0,1
while f2<1000:
      print(f2)
      f1,f2 = f2,f1+f2
```

Una vez corrido el programa podemos darnos cuenta como es que trabaja; el ciclo comienza con las cantidades especificadas antes de la condición, estos serán $f1_0$ y $f2_0$, con ellos se realizan la primera operación, la cual en este caso seria tomas a $f2$ como la nueva $f1$, para después realizar la operación de la suma; nuevamente, se toma el resultado de $f2$ y se guarda en la memoria como $f1$ y en con los resultados anteriores que se realiza la suma.

A partir de la comprensión de este tipo de programas y de como debemos de utilizar los ciclos es posible realizar otro tipo de sucesiones, por ejemplo, el siguiente programa nos muestra los números de la seria de Catalan que son menores a 1,000,000:

```
print("-------------------- :)")
print("Secuencia de Catalan.")
n,C = 0.,1.
while C<=1000000:
    print(C)
    n,C = n+1,((2*(2*n+1))/(n+2))*C
```

Podemos darnos cuenta, que las diferencias con el primer programa son pequeñas, por ejemplo fue necesario agregar un punto después de los números

iniciales de n y C ya que Python necesita que se le especifiquen si las operaciones se realizaran con números reales para que nos muestre resultados reales; de otro modo, el programa trabaja únicamente con números enteros y esto altera el resultado.

Además de eso, se debe de señalar que la cantidad de n irá aumentando en una cifra cada vez, lo que resuelve con una simple suma; para finalmente especificar la operación que debe realizarse para obtener C. De esta forma se obtiene el siguiente resultado:

```
-------------------- :)
Secuencia de Catalan.
1.0
1.0
2.0
5.0
14.0
42.0
132.0
429.0
1430.0
4862.0
16796.0
58786.0
208012.0
742900.0
```

Referencias

[1] Wikipedia, The free encyclopedia; "Python"; 2016

[2] Wikipedia, The free encyclopedia; "Spherical coordinate system"; 2016