Facultad de Matemática, Astronomía y Física, U.N.C.

Computación (profesorados en física y matemática)

Guía N° 3 (2019)

Problema 1: La fórmula para convertir grados Fahrenheit a grados Celsius es

$$C = \frac{5}{9}(F - 32).$$

Escriba una función de Python `C(F)` que implemente esta fórmula y otra para la función inversa.

Problema 2: Escriba un programa que pida una temperatura y su unidad correspondiente (grados F o C) y, usando las funciones anteriores imprima el valor de la tempratura en la otra unidad. Ayuda: use la sentencia `if`.

Problema 3: Escriba una función que calcule la altura y la velocidad de un problema de tiro vertical en el tiempo t provisto por el usuario, dadas una velocidad y altura inicial. Recuerde que

$$y(t) = -\frac{1}{2}gt^2 + v_0 t + y_0.$$

Problema 4: Escriba una función `raiz(a,b,c)` para resolver cualquier ecuación de la forma:

$$ax^2 + bx + c = 0.$$

La función debe retornar las raices de la ecuación, las cuales deben ser objectos `float` cuando las raices son reales, u objetos `complex` cuando no lo son. Pruebe su función con distintos casos con soluciones conocidas con raíces reales y complejas.

Problema 5: Sea $f : [a, b] \to \mathbb{R}$ una función contínua tal que $f(a)f(b) < 0$. Escriba un programa en Python, en forma de script, que implemente el método de bisección para hallar una raíz $r \in (a, b)$ de $f(x)$. El programa debe detenerse cuando la aproximación x_n de r obtenida al cabo de n iteraciones satisfaga las condiciones: $|x_n - r| < \delta$ y $|f(x_n)| < \varepsilon$, donde $\varepsilon > 0$ y $\delta > 0$ son tolerancias dadas. En este caso el programa debe imprimir en pantalla la aproximación a la raíz y el número de iteraciones realizadas. Si al cabo de M iteraciones la aproximación obtenida no satisface la tolerancia deseada, el programa debe detenerse indicando que no se obtuvo una aproximación satisfactoria. Antes de iniciar la iteración el programa debe verificar la condición inicial $f(a)f(b) < 0$, si esta no se cumple, el programa de terminar con un mensaje de error que indique esta situación.

Utilizando $M = 100$, $\varepsilon = 10^{-6}$, y $\delta = 10^{-6}$, calcule raíces positivas de las siguientes funciones e intervalos

a) $f(x) = 3x - \tan(x)$, en $[0.1, 1.5]$.

b) $f(x) = x^6 - 5x^5 + x^4 - 3x^3 + x^2 - x + 1$, en $[0, 1]$.

Problema 6: Un triángulo arbitrario puede ser descripto por las coordenadasde sus tres vértices (x_1, y_1), (x_2, y_2), (x_3, y_3), numerados en sentido antihorario. El área del triángulo está dada por la fórmula:

$$A = \frac{1}{2}|x_2y_3 - x_3y_2 + x_3y_1 + x_1y_2 - x_2y_1|$$

Escriba una función `area(vertices)` que retorne el área de un triángulo cuyos vértices se especifiquen por el argumento `vertices`, el cual debe ser una lista anidad de las coordenadas de

los vértices. Por ejemplo, el cómputo del area de un triángulo con vértices $(0,0),(1,0)$, y $(0,2)$ debe realizarse de la siguiente manera:

```
triangulo1 = area([[0,0],[1,0],[0,2]])
```

o bien con

```
v1 = (0,0); v2 = (1,0); v3 = (0,2);
vertices = [v1, v2, v3]
triangulo1 = area(vertices)

print 'El area del triangulo es %.2f' % triangulo1
```

Problema 7: Computando la longitud de un camino. Un objeto se mueve a lo largo de un camino en un plano. Hemos grabado la posición (x, y) del mismo en n instantes de tiempos: $(x_0, y_0), (x_1, y_1), \ldots (x_{n-1}, y_{n-1})$. El largo total L del camino desde (x_0, y_0) hasta (x_{n-1}, y_{n-1}) puede aproximarse por la suma de los largos de segmentos rectilíneos, es decir

$$L = \sum_{i=1}^{n-1} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$$

Haga una función `long_de_camino(x,y)` para computar L de acuerdo a la fórmula anterior. Los argumentos `x` e `y` deben contenter las coordendas $x_0, x_1, \ldots, x_{n-1}$ y $y_0, y_1, \ldots, y_{n-1}$. Pruebe la función en un camino triangular con cuatro puntos $(1,1),(2,1),(1,2) y (1,1)$

Problema 8: Utilice su función `long_de_camino` para calcular π. Para ello recuerde que π es igual al perímetro de un círculo de radio $1/2$. Suponga que aproxima a la circunferencia por un polígono de $N+1$ vértices sobre el círculo. La longitud de éste polígono puede encontrarse usando la función `long_de_camino`. Note que puede generar $N+1$ puntos sobre el círculo con las fórmulas:

$$x_i = \frac{1}{2}\cos\Big(\frac{2\pi i}{N}\Big), \qquad y_i = \frac{1}{2}\sin\Big(\frac{2\pi i}{N}\Big), \qquad i = 0, 2, 3, \ldots, N$$

Problema 9: Una manera de aproximar numéricamente la derivada de una función es mediante el cociente incremental centrado, es decir

$$f'(x) \simeq \frac{f(x+h) - f(x-h)}{2h}$$

donde h es pequeño.

Escriba una función de Python `derivada(f,x,h=1e-6)` que retorne la aproximación anterior de la derivada de una función matemática representada por una función de Python `f(x)`.

Problema 10: Escriba dos funciones `vmax(a)` y `vmin(a)` que sean capaces de encontrar los valores máximo y mínimo de la lista `a`. Para hacerlo utilice la siguiente técnica: inicialice una variable `amax` con el primer elemento de la lista, entonces recorra todos los elementos restantes de `a` (`a[1:]`); compare cada elemento con `amax`, y si es más grande, haga que amax guarde ese valor. Use una técnica similar para computar el valor mínimo.

Problema 11: Explique la diferencia entre `if` y `elif`. Considere el siguiente ejemplo:

```
def where1(x,y):
    if x > 0:
```

```
        print 'Cuadrante I o IV'
    if y > 0:
        print 'Cuadrante I o II'

def where2(x,y):
    if x > 0:
        print 'Cuadrante I o IV'
    elif y > 0:
        print 'Cuadrante II'

for x, y in (-1, 1), (1, 1):
    where1(x,y)
    where2(x,y)
```

¿Qué es lo que se imprime? Explique el funcionamiento del programa anterior.

Problema 12: Usando instrucciones `if / elif` haga un script que convierta los nombres de las notas musicales "DO, RE, MI, FA, Sol, LA, SI" al sistema de notación musical inglés: "C,D,E,F,G,A,B". Su programa debe contener un buble while infinito y esperar que el usuario ingrese por teclado el nombre de la nota. La correspondiente note del sistema inglés debe imprimirse en pantalla inmediatamente después de ingresar la nota.

Problema 13: Las notas musicales son DO, RE, MI, FA, Sol, LA y SI, repitiéndose en esa secuencia. La diferencia entre ellas es un tono excepto entre MI y FA, y SI y DO que corresponden a un semitono. Las notas además se relacionan con frecuencias determinadas de vibración del aire de manera que para pasar a un semitono más agudo hay que multiplicar la frecuencia de la nota de referencia por $2^{1/12}$. Además el LA de referencia para afinación corresponde a 440Hz. Haga una función a la cual se le ingrese una dada frecuencia y devuelva la nota que le corresponde y además diga si corresponde a una nota afinada (error menor a 1Hz).

Problema 14: Sea $f(x)$ una función continuamente diferenciable. Escriba un programa en Python, en forma de script, que implemente el método de Newton para hallar raíces simples de f. La iteración de Newton debe detenerse cuando $|x_n - x_{n-1}| < \delta$ y $|f(x_n)| < \varepsilon$. Si estas tolerancias no son alcanzadas al cabo de M iteraciones, el programa debe detenerse indicando que no se alcanzó la tolerancia deseada. Si el programa alcanza las tolerancias deseadas, al detenerse debe imprimir en pantalla el valor de la aproximación x_n, el valor de $|f(x_n)|$ y el número de iteraciones utilizadas.

Usando $M = 20$, $\delta = \varepsilon = 10^{-6}$, halle las raices de

a) $f(x) = 3x - \tan(x)$, usando $x_0 = 1.5$.

b) $f(x) = x^6 - 5x^5 + x^4 - 3x^3 + x^2 - x + 1$, usando $x_0 = 0$.

Problema 15: Repita el problema anterior pero ahora usando el método de la Secante. Los criterios de detención son idénticos a los de Newton.

Usando $M = 40$, $\delta = \varepsilon = 10^{-6}$, halle las raices de

a) $f(x) = 3x - \tan(x)$, usando $x_0 = 0.1$ y $x_1 = 1.5$

b) $f(x) = x^6 - 5x^5 + x^4 - 3x^3 + x^2 - x + 1$, usando $x_0 = 0$ y $x_1 = 1$.