INTRODUCCIÓN A PYTHON.
EJERCICIOS CLASE 1

OSCAR S. DALMAU CEDEÑO
CENTRO DE INVESTIGACIÓN EN MATEMÁTICAS, CIMAT A.C.

1. Ejercicios. Elementos de Programación

(1) Almacenar en una lista las raíces cuadradas de los primeros 100 números múltiplos de 6.
(2) Almacenar en una lista los productos entre los 20 primeros números impares con los 20 primeros números que son múltiplos de 4.
(3) Decidir si un número natural n es primo.
(4) La Sucesión de Fibonacci empieza con los números 0 y 1, y los términos restantes son la suma de los dos términos anteriores

$$\begin{aligned} f_1 &= 0; \\ f_2 &= 1; \\ f_n &= f_{n-1} + f_{n-2},\ n = 3, 4, \cdots \end{aligned}$$

a) Definir una función que determine el enésimo término de la serie.
(5) Escriba un script que lea los coeficientes de las rectas

$$ax + by = c$$
$$dx + ey = f$$

y determine si son paralelas, y en caso de no serlo, determine si son perpendiculares.
(6) Calcular la suma de los n primeros términos de la sucesión formada por los siguientes elementos 1, $\frac{x}{1!}$, $\frac{x^2}{2!}$, $\frac{x^3}{3!}, \ldots,$
(7) Calcular la suma de los n primeros términos de la sucesión formada por las derivadas de la función: $f(x) = x^n, n \in \mathbb{N}$
(8) Implementar una función en que permita calcular la raíz n-ésima de un número. Para ello implementar un algoritmo iterativo que use la siguiente fórmula

$$x_{t+1} = \frac{1}{n}\left(\frac{A}{x_t^{n-1}} + (n-1)x_t\right)$$

```
def r = raizn(x,n, maxIter, tol):
...
...  return r
```

En la función anterior

(a) t Representa el número de iteración.
(b) x Es un número real positivo
(c) n Es un número natural y representa el orden de la raíz
(d) $maxIter$ Es el número máximo de iteraciones del algoritmo (*criterio de paro*).
(e) tol Es la tolerancia del error para el *criterio de paro*, es decit $|x_{t+1} - x_t| < tol$, $|x_{t+1} - x_t|/|x_{t+1}| < tol$)

(9) Implementar las siguientes reglas de integración numérica

(a) Regla de los trapecios:

$$\int_a^b f(x)dx \approx \frac{h}{2}\left(f(a) + f(b)\right) + h\sum_{k=1}^{n-1} f(x_k)$$

donde $h = (b-a)/n$, $x_k = a + kh,\ k = 0, 1, \ldots, n$

(b) Regla de Simpson:

$$\int_a^b f(x)dx \approx \frac{h}{3}\left(f(a) + f(b)\right) + \frac{2h}{3}\sum_{k=1}^{n-1} f(x_{2k}) + \frac{4h}{3}\sum_{k=1}^{n} f(x_{2k-1})$$

donde $h = (b-a)/(2n)$, $x_k = a + kh,\ k = 0, 1, \ldots, 2n$

(10) El conjunto de Mandelbrot se construye mediante la siguiente recursión:

$$\begin{aligned} z_0 &= 0 \\ z_{n+1} &= z_n^2 + c,\ n = 0, 1, \cdots, N \end{aligned}$$

donde $c \in \mathbb{C}$. Muestre conjuntos de Mandelbrot para c, N dado y $|z_n| \leq 2$.

- Sugerencia considera $c = x + \mathrm{i}y$ con $x, y \in [-2, 2]$. Tomar una discretización del intevalo $[-2, 2]$ y grafica H donde

$$\begin{aligned} H &= \frac{W}{\max(W)} \\ W &= \frac{1}{1 + |z_{N+1}|} \end{aligned}$$

Nota: z_{N+1} es una matriz, y por tanto H se puede mostrar como una imagen.

(11) Escriba script que permita mostrar el triángulo de Sierpinski. La entrada del algoritmo son las coordenadas de los vertices de un triangulo y el numero de veces que se repite el algoritmo.

Paso 1: Se muestran los vertices de un triángulo T.
Paso 2: Se crea un nuevo triangulo cuyos vértices son los puntos medios del triángulo T.
Paso 3: De los cuatro triángulo que se han formado se descarta el triángulo central.

Paso 4: Para cada uno de los tres triángulo restantes se aplican a los pasos 1, 2 y 3.