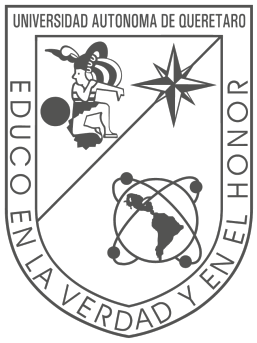
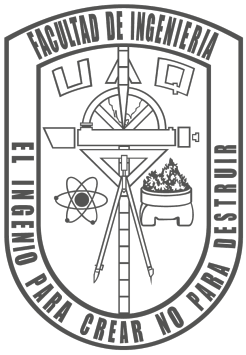
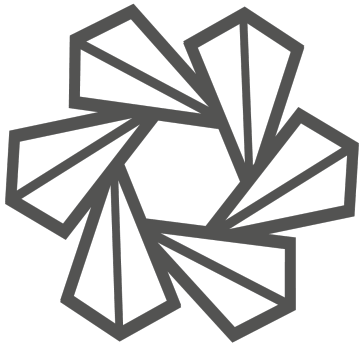
Universidad Autónoma de Querétaro

Facultad de Ingeniería
División de Investigación y Posgrado







Reporte 1

Ejercicios. Elementos de Programación

Maestría en Ciencias en Inteligencia Artificial
Optativa de especialidad II - Deep Learning

Aldo Cervantes Marquez
Expediente: 262775
Profesor: Dr. Sebastián Salazar Colores

Santiago de Querétaro, Querétaro, México
Semestre 2022-2
8 de Agosto de 2022

Índice

1. Introducción

La presente práctica consta de 6 ejercicios de programación en el lenguaje de alto nivel Python, en donde se aplicarán los conceptos básicos de programación en este lenguaje.

Esta práctica permitirá sentar y refrescar las bases de los principios fundamentales de la programación de Python para futuros programas que permitan desarrollar tareas más especificas y enfocadas a la inteligencia artificial.

2. Justificación

El uso de lenguaje Python ha sido ampliamente utilizado a lo largo del mundo y ha adquirido mucha popularidad por su versatilidad [1] pues se ha convertido en uno de los tres lenguajes más usados actualmente. Esto motiva más a conocer y aprender estos lenguajes que permiten desarrollar programas y aplicaciones variadas. En este caso, el lenguaje Python se ha utilizado mucho para aplicaciones de inteligencia artificial y técnicas de aprendizaje automático [2]. Por lo que esta práctica se encuentra sustentada en el aprendizaje y destreza del lenguaje Python para irse familiarizando con las diferentes funciones y capacidades del lenguaje.

3. Conclusiones

Los ejercicios fueron de mucha utilidad para recordar y aplicar algunos conceptos que ya tenía mucho que no usaba. Considero que aprendí y apliqué de manera correcta los siguientes conceptos:

- Uso y operaciones con listas
- Recursividad
- Uso de bucles y ciclos

Estos conceptos fueron reforzados, estudiados y aplicados con ayuda de una plataforma educativa para aprender a programar [3]. Finalmente considero que estos ejercicios me servirán como bases para entender códigos mas complejos y basados en inteligencia artificial, pues conocer esto es algo básico para avanzar en los temas de interés. Se anexa el archivo con la extensión correspondiente.

Referencias

[1] Tokio-New-Technology-School, "Inteligencia Artificial con Python: los secretos." https://www.tokioschool.com/noticias/algoritmos-inteligencia-artificial-python/, 2021.

[2] R. KeepCoding, "Python, el lenguaje de programación más popular | KeepCoding Tech School." https://keepcoding.io/aprender/python-lenguaje-de-programacion-mas-popular/, Jan. 2021.

[3] W3Schools, "Python Tutorial." https://www.w3schools.com/python/, 2022.

Anexo: Código en Google Colab

August 8, 2022

1 Ejercicio 1

Almacenar en una lista las raíces cuadradas de los primeros 100 números múltiplos de 6.

Para la solución de este ejercicio se realizó un ciclo finito para obtener los primeros 100 múltiplos de 6 (multiplicando por 6 cada numero de la secuencia) y obteniendo sus raices mientras se almacenan con la funcion `append()` que coloca cada valor en la última posición de la lista

[2]:
```
import math
import numpy as np

lista=[]

for x in range(1,101):

    lista.append(math.sqrt(x*6))


print(lista)
```

```
[2.449489742783178, 3.4641016151377544, 4.242640687119285, 4.898979485566356,
5.477225575051661, 6.0, 6.48074069840786, 6.928203230275509, 7.3484692283495345,
7.745966692414834, 8.12403840463596, 8.48528137423857, 8.831760866327848,
9.16515138991168, 9.486832980505138, 9.797958971132712, 10.099504938362077,
10.392304845413264, 10.677078252031311, 10.954451150103322, 11.224972160321824,
11.489125293076057, 11.74734012447073, 12.0, 12.24744871391589,
12.489995996796797, 12.727922061357855, 12.96148139681572, 13.19090595827292,
13.416407864998739, 13.638181696985855, 13.856406460551018, 14.071247279470288,
14.2828568570857, 14.491376746189438, 14.696938456699069, 14.89966442575134,
15.0996688705415, 15.297058540778355, 15.491933384829668, 15.684387141358123,
15.874507866387544, 16.06237840420901, 16.24807680927192, 16.431676725154983,
16.61324772583615, 16.792855623746664, 16.97056274847714, 17.146428199482248,
17.320508075688775, 17.4928556845359, 17.663521732655695, 17.832554500127006,
18.0, 18.16590212458495, 18.33030277982336, 18.49324200890693,
18.65475810617763, 18.81488772222678, 18.973665961010276, 19.131126469708992,
19.28730152198591, 19.44222209522358, 19.595917942265423, 19.748417658131498,
19.8997487421324, 20.049937655763422, 20.199009876724155, 20.346989949375804,
20.493901531919196, 20.639767440550294, 20.784609690826528, 20.92844953645635,
21.071307505705477, 21.213203435596427, 21.354156504062622, 21.494185260204677,
```

```
21.633307652783937, 21.77154105707724, 21.908902300206645, 22.045407685048602,
22.181073012818835, 22.315913604421397, 22.44994432064365, 22.58317958127243,
22.715633383201094, 22.847319317591726, 22.978250586152114, 23.108440016582687,
23.2379000772445, 23.366642891095847, 23.49468024894146, 23.62202362203543,
23.748684174075834, 23.874672772626646, 24.0, 24.124676163629637,
24.24871130596428, 24.372115213907882, 24.49489742783178]
```

2 Ejercicio 2

Almacenar en una lista los productos entre los 20 primeros números impares con los 20 primeros números de son múltiplos de 4.

Para solucionar este ejercicio se realizó de igual manera un bucle finito para obtener cada valor para finalmente multiplicarlos con el comando `np.multiply()` que permite realizar la multiplicación de listas elemento a elemento y finalmente se aplicó la función `tolist()` que permite convertir el resultado en variable de tipo lista.

[3]:
```
#inicio de programa 2
impares=[]
m_4=[]
lista2=[]
for x in range(0,20):
    impares.append((2*x)+1)
    m_4.append((x+1)*4)

print('Lista de 20 números impares',impares)
print('Lista de 20 números múltiplos de 4',m_4)
lista2=np.multiply(impares,m_4)
lista2=lista2.tolist()
print('Lista multiplicada',lista2)
```

```
Lista de 20 números impares [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27,
29, 31, 33, 35, 37, 39]
Lista de 20 números múltiplos de 4 [4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44,
48, 52, 56, 60, 64, 68, 72, 76, 80]
Lista multiplicada [4, 24, 60, 112, 180, 264, 364, 480, 612, 760, 924, 1104,
1300, 1512, 1740, 1984, 2244, 2520, 2812, 3120]
```

3 Ejercicio 3

Decidir si un numero natural n es primo.

Para la solución de este ejercicio se proponen 2 soluciones alternas. La primera (3) propone que al introducir el número se haga un barrido por todos los números entre 2 y el número. En caso de encontrar que algún valor sea divisible (modulo igual a 0), entonces la variable $flag$ cambia de valor y sale del bucle for para imprimir que el número no es primo. En caso contrario, es decir que una vez terminado el barrido no haya encontrado ningún valor divisible, entonces la variable manda imprimir que el número si es primo.

Para la solucion 3.1, es algo muy parecido, con la única diferencia de que al momento de encontrar algún valor divisible, imprime que no es número primo y forza a terminar el programa.

[4]:
```
#inicio de programa 3
a=int(input("Introduzca número para evaluar si es primo: "))
flag=0
for x in range (2,a):

    if a%x==0:
        flag=1
        break

if flag==0:
    print("Si es número primo")
else:
    print("No es número primo")
#Variante del programa 3.1

#import sys
#a=int(input())
#for x in range (2,a):
#    if a%x==0:
#        print("No es número primo")
#        sys.exit(1)

#print("Si es número primo")
```

```
Introduzca número para evaluar si es primo: 281
Si es número primo
```

4 Ejercicio 4

La sucesión de Fibonacci empieza con los numeros 0 y 1, y los términos restantes son la suma de los dos términos anteriores:

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

a) Definir una función que determine el enésimo término de la serie.

Para la solución de este problema, se ocupa el concepto de recursividad, en donde, con base a las formulas dadas, se puede volver a llamar a la función. se utilizó el comando `def ___():`

[5]:
```
def fibonacci(n):
    if n==0:
        return 0
    elif n==1:
        return 1
```

```
    else:
        return fibonacci(n-1)+fibonacci(n-2)

n_f=int(input("Introduzca el número de Fibonacci para calcular "))
print("El resultado es: ",fibonacci(n_f))
```

```
Introduzca el número de Fibonacci para calcular 9
El resultado es:  34
```

5 Ejercicio 5

Escriba un script que lea los coeficientes de las rectas

$$\begin{aligned} ax + by &= c \\ dx + ey &= f \end{aligned} \tag{2}$$

y determine si son paralelas, y en caso de no serlo determine si son perpendiculares.

Para solucionar este problema se debe calcular la pendiente m (función pen y tab) y posteriormente multiplicarlas (evaluar) de tal modo que:

$$\begin{aligned} m_1 * m_2 = -1 &\therefore perpendicular \\ m_1 = m_2 &\therefore paralela \end{aligned} \tag{3}$$

[6]:
```
def tab(a,b,c):
    return [3,5,(c-a*3)/b,(c-a*5)/b]

def pen(y):
    return ((y[3]-y[2])/(y[1]-y[0]))

def evaluar(m1,m2):
    if m1==m2:
        return "Son rectas paralelas"
    elif m1*m2==-1:
        return "Son rectas perpendiculares"
    else:
        return "No son rectas paralelas ni perpendiculares"

a=float(input("Introduzca coeficiente de a: "))
b=float(input("Introduzca coeficiente de b: "))
c=float(input("Introduzca coeficiente de c: "))
d=float(input("Introduzca coeficiente de d:"))
e=float(input("Introduzca coeficiente de e: "))
f=float(input("Introduzca coeficiente de f: "))

r_1=tab(a,b,c)
r_2=tab(d,e,f)
```

```
m1=pen(r_1)
m2=pen(r_2)
print(evaluar(m1,m2))
```

```
Introduzca coeficiente de a: 1
Introduzca coeficiente de b: 1
Introduzca coeficiente de c: 1
Introduzca coeficiente de d:-1
Introduzca coeficiente de e: 1
Introduzca coeficiente de f: 1
Son rectas perpendiculares
```

Se utilizó con los siguientes ejemplos:

Rectas paralelas:

$$\begin{aligned} 2x+y &= 1 \\ 2x+y &= 3 \end{aligned} \tag{4}$$

Rectas perpendiculares:

$$\begin{aligned} x+y &= 1 \\ -x+y &= 1 \end{aligned} \tag{5}$$

6 Ejercicio 6

Calcular la suma de los n primeros términos de la sucesión formada por los siguientes elementos $1, \frac{x}{1!}, \frac{x^2}{2!}, \frac{x^3}{3!}, ...,$

Para la solución de este ejercicio se realizó un bucle finito en el que se irá desde el primer valor de x=1 hasta el deseado, evaluado en el valor deseado. realizando una suma de la misma variable al igualarla a si misma adicionada al siguiente número de la sucesión. Finalmente se suma el primer valor de la sucesion en n=0 que es 1. Esta función aproxima numéricamente a la expresión e^x

[7]:
```
def suma(n,x):
    total=0
    for a in range(1,n):
        total=total+ (x**a/math.factorial(a))
    total=total+1
    return total

n=int(input("Introduzca el número de términos de la sucesión: "))
x=float(input("Introduzca el valor de x a ser evaluado: "))

print("El valor de la suma de la sucesión es: ",suma(n,x))
```

```
Introduzca el número de términos de la sucesión: 6
Introduzca el valor de x a ser evaluado: 0.5
El valor de la suma de la sucesión es:  1.6486979166666667
```