TALLER INTERPOLACION

Andres Porras , Alejandro Diaz, Cristian Benitez

April 2021

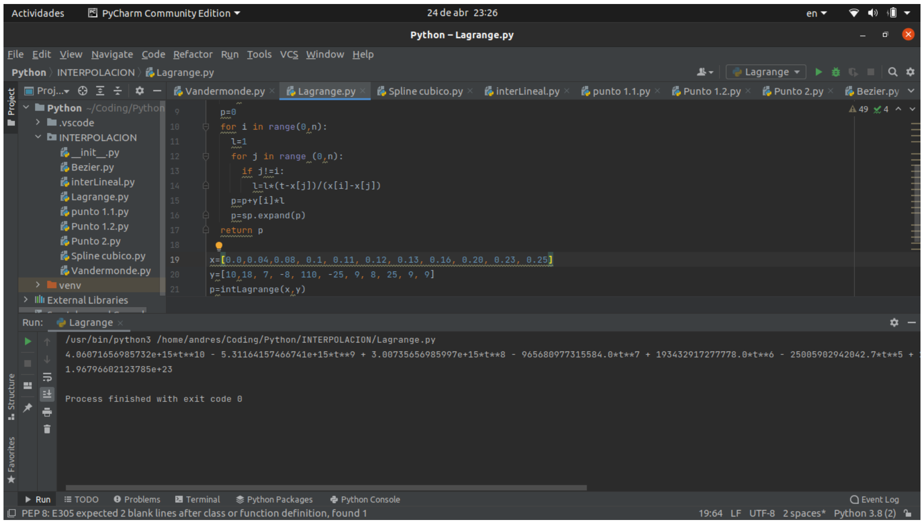
1. Ejercicio 1

Demuestre que dados los n + 1 puntos distintos (xi, yi) de una funcion definida y continua en [a, b] el polinomio interpolante que incluye a todos los puntos es unico.

- Solucion Teniendo en cuenta el algoritmo del metodo de interpolacion de lagrange;

```
    import sympy as sp
def intLagrange(x,y,u=None):
  n=len(x)
  if u==None:
    t=sp.Symbol('t')

  else:
    t=u
  p=0
  for i in range(0,n):
    l=1
    for j in range (0,n):
      if j!=i:
        l=l*(t-x[j])/(x[i]-x[j])
    p=p+y[i]*l
    p=sp.expand(p)
  return p

x=[0.0,0.04,0.08, 0.1, 0.11, 0.12, 0.13, 0.16, 0.20, 0.23, 0.25]
y=[10,18, 7, -8, 110, -25, 9, 8, 25, 9, 9]
p=intLagrange(x,y)
print (p) #Polinomio de interpolacion
a=intLagrange(x,y,6) #Evaluar el polinomio en otro punto
print(a)
```

Nos da como salida los siguientes resultados:

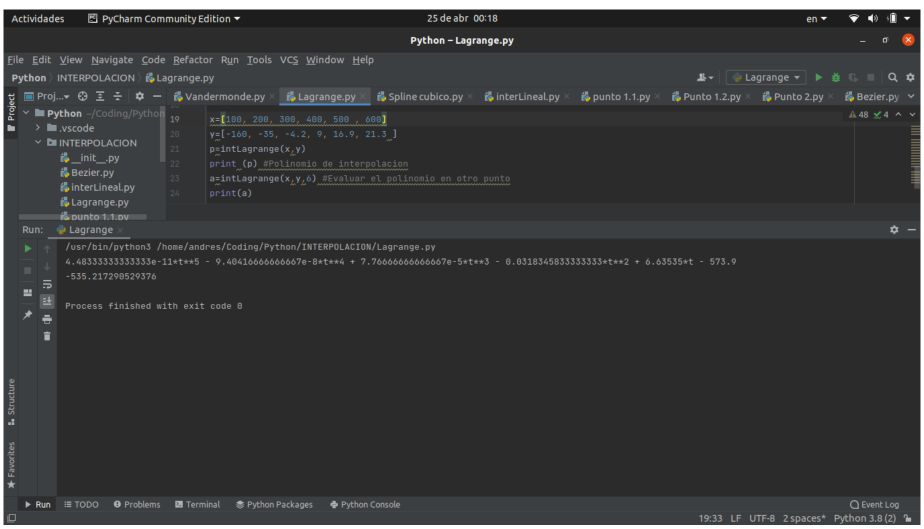


Ahora haciedno la prueba en los puntos:

x=[100, 200, 300, 400, 500 , 600] y=[-160, -35, -4.2, 9, 16.9, 21.3]

Evaluados en un otro punto = 6

Obtuvimos los siguientes resultados:



Donde se puede determinar que los polinomios generados son unicos siempre y cuando los puntos cumplan con los parametros.

2. Ejercicio 10

Considere el comportamiento de gases no ideales se describe a menudo con la ecuacion virial de estado. los siguientes datos para el nitrogeno N2.

T(K)	100	200	300	400	450	500	600
$B(cm^3)/mol$	-160	-35	-4.2	9.0		16.9	21.3

Donde T es la temperatura [K] y B es el segundo coeficiente virial. El comportamiento de gases no ideales se describe a menudo con la ecuaci´on virial de estado.

$$\frac{PV}{RT} = 1 + \frac{B}{V} + \frac{C}{V^2} + ...,$$

Donde P es la presion, V el volumen molar del gas, T es la temperatura Kelvin y R es la constante de gas ideal.Los coeficientes B = B(T), C = C(T), son el segundo y tercer coeficiente virial, respectivamente. En la practica se usa la serie truncada para aproximar.

$$\frac{PV}{RT} = 1 + \frac{B}{V}$$

a) Determine un polinomio interpolante para este caso b) Utilizando el resultado anterior calcule el segundo y tercer coeficiente virial a 450K. c) Grafique los puntos y el polinomio que ajusta d) Utilice la interpolacion de Lagrange y escriba el polinomio interpolante e) Compare su resultado con la serie truncada (modelo teorico), cual aproximacion es mejor por que?

- Solucion

  Para realizacion de este ejercicio, tuvimos en cuenta el algoritmo de dos metodos de interpolacion:

  METODO DE LAGRANGE

```
    import numpy as np
import sympy as sp


def intLagrange(x, y, u=None):
    n = len(x)
    if u == None:
        t = sp.Symbol('t')
```

```
    else:
        t = u
    p = 0
    for i in range(0, n):
        l = 1
        for j in range(0, n):
            if j != i:
                l = l * (t - x[j]) / (x[i] - x[j])
        p = p + y[i] * l
        p = sp.expand(p)
    return p


x = [0.0,0.04,0.08, 0.1, 0.11, 0.12, 0.13, 0.16, 0.20, 0.23, 0.25]
y = [10,18, 7, -8, 110, -25, 9, 8, 25, 9, 9]
p = intLagrange(x, y)

print(p)  # Polinomio de interpolacion
a = intLagrange(x, y, 450)  # Evaluar el polinomio en otro punto
print(a)

# #Gr fica
plt.plot(x, y, 'o', label='Puntos')
plt.legend()
plt.xlabel('x')
plt.ylabel('Y')
plt.title('Interpolaci n Lagrange')
plt.show()
```

METODO DE DIFERENCIAS DIVIDIDAS - NEWTON

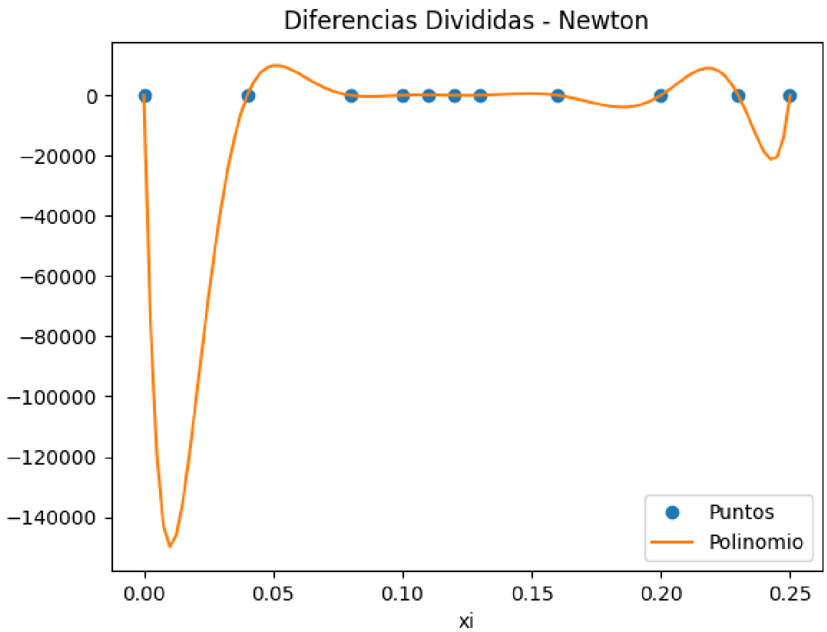
```
    import numpy as np
import sympy as sym
import matplotlib.pyplot as plt

xi = np.array([0.0,0.04,0.08, 0.1, 0.11, 0.12, 0.13, 0.16, 0.20, 0.23, 0.25]
fi = np.array([10,18, 7, -8, 110, -25, 9, 8, 25, 9, 9])

# PROCEDIMIENTO

# Tabla de Diferencias Divididas Avanzadas
titulo = ['i   ','xi  ','fi  ']
n = len(xi)
ki = np.arange(0,n,1)
tabla = np.concatenate(([ki],[xi],[fi]),axis=0)
tabla = np.transpose(tabla)
```

```
# diferencias divididas vacia
dfinita = np.zeros(shape=(n,n),dtype=float)
tabla = np.concatenate((tabla,dfinita), axis=1)

# Calcula tabla, inicia en columna 3
#tama o de la matriz
[n,m] = np.shape(tabla)
diagonal = n-1
j = 3
while (j < m):
    # cada fila de columna
    i = 0
    paso = j-2 # inicia en 1
    while (i < diagonal):
        denominador = (xi[i+paso]-xi[i])
        numerador = tabla[i+1,j-1]-tabla[i,j-1]
        tabla[i,j] = numerador/denominador
        i = i+1
    diagonal = diagonal - 1
    j = j+1

# POLINOMIO con diferencias Divididas
# caso: puntos equidistantes en eje x
dDividida = tabla[0,3:]
n = len(dfinita)



x = sym.Symbol('x')
polinomio = fi[0]
for j in range(1,n,1):
    factor = dDividida[j-1]
    termino = 1
    for k in range(0,j,1):
        termino = termino*(x-xi[k])
    polinomio = polinomio + termino*factor

# simplifica multiplicando entre (x-xi)
polisimple = polinomio.expand()

# polinomio para evaluacion num rica
px = sym.lambdify(x,polisimple)

# Puntos para la gr fica
muestras = 101
a = np.min(xi)
```

```
b = np.max(xi)
pxi = np.linspace(a,b,muestras)
pfi = px(pxi)


print('polinomio con Newton: ' )
print(polisimple)
solucion = px(450)
print(solucion)



plt.plot(xi,fi,'o', label = 'Puntos')
plt.plot(pxi,pfi, label = 'Polinomio')
plt.legend()
plt.xlabel('xi')
plt.ylabel('fi')
plt.title('Diferencias Divididas - Newton')
plt.show()
```

En donde ambos metodos de interpolacion determinan el mismo resultado el cual es: 1.3786853674743429e+42

Gracias a la libreria de matplot, logramos representar los puntos y obtuvimos el siguiente grafico:



3. Ejercicio 13

Escala de gravamen del Impuesto a la renta

Base imponible	Cuota íntegra	Tipo
4.410.000	1.165.978	38,86%
4.830.000	1.329.190	41,02%
5.250.000	1.501.474	43,18%
5.670.000	1.682.830	

La cuota íntegra del Impuesto sobre la Renta se determina aplicando una fórmula basada en la interpolación lineal. Un contribuyente tiene una base imponible de 5 millones. Para calcular lo que tiene que pagar a Hacienda efectúa las siguientes operaciones, consultando la escala de gravamen anterior:

Base	5.000.000		Cuota
Hasta	4.830.000		1.329.190
	Resto.... 170.000	al 41,02%	69.734
		SUMA	1.398.924

El tipo marginal del 41,02 porciento que aparece en la escala de gravamen es precisamente el cociente de las diferencias entre las cuotas íntegras y las bases imponibles más próximas en la escala a los 5 millones.

$$\frac{1.501.474 - 1.329.190}{5.250.000 - 4.830.000} = 0.4102$$

La fórmula aplicada es, en definitiva;

Cuota = 1.329.190 + 0,4102(Base 4.830.000)

Para las bases comprendidas en el intervalo [4.830.000,5.250.000].

En particular, para una base imponible de 5.250.000 es indiferente aplicar la fórmula anterior o tomar directamente el valor de la tabla. En términos matemáticos esto equivale a decir que la Cuota es una función continua de la Base imponible. El Impuesto sobre la Renta es progresivo, es decir, que el tipo de la imposición aumenta con la base imponible, como se comprueba observando la escala de gravamen. Así, el tipo medio correspondiente a 4.830.000 es el 27,52pociento y el de 5.250.000 es el 28,60 porciento.

El contribuyente se siente perjudicado por el hecho de que al Resto de su Base imponible (170.000) se le aplica el mismo tipo marginal (41,02 porciento) que, a otro contribuyente con una Base de 5.250.000, alegando que debe aplicársele el correspondiente a la base más próxima en la escala (4.830.000) que es del 38,86. Hacienda, por su parte, rechaza estos argumentos y efectúa la liquidación según sus normas. El sujeto del impuesto interpone recurso (tutela) ante el Tribunal competente, que considera en parte sus alegaciones. El fallo establece que en todo caso se debería aplicar un tipo marginal intermedio. Como experto en temas fiscales debes elaborar un informe para que Hacienda conozca las diferencias entre el actual sistema impositivo y los posibles métodos de determinar la imposición correspondiente a la base de 5 millones por interpolación de segundo y tercer grado en la escala de gravamen. ¿En cada grado debe añadirse la base más próxima a 5 millones?

- Solucion Teniendo en cuenta los puntos a validar y el tercer valor el cual era '5,000,000', se procedio a hacer la comprobacion con tres metodos de interpolacion los cuales fueron;

  DIFERENCIAS DIVIDAS - NEWTON

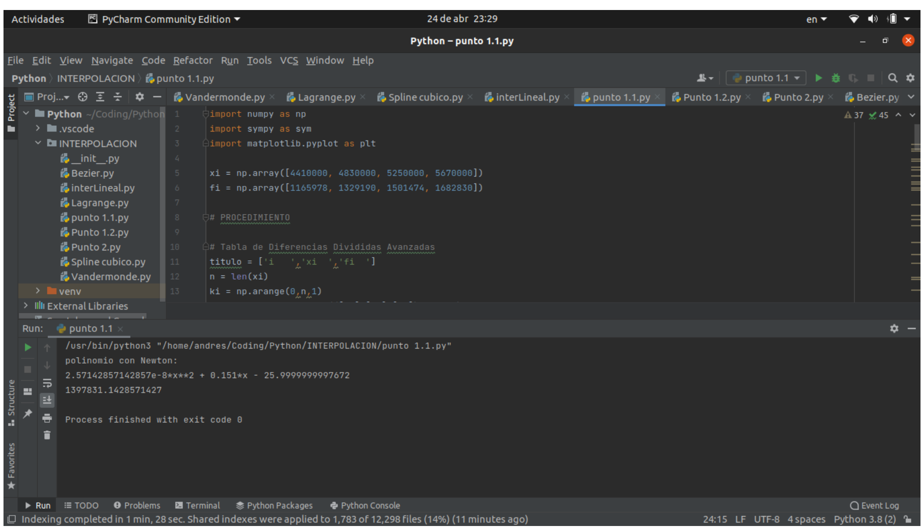
```
    import numpy as np
import sympy as sym
import matplotlib.pyplot as plt

xi = np.array([4410000, 4830000, 5250000, 5670000])
fi = np.array([1165978, 1329190, 1501474, 1682830])
```

  # PROCEDIMIENTO

```
# Tabla de Diferencias Divididas Avanzadas
titulo = ['i    ','xi   ','fi   ']
n = len(xi)
ki = np.arange(0,n,1)
tabla = np.concatenate(([ki],[xi],[fi]),axis=0)
tabla = np.transpose(tabla)

# diferencias divididas vacia
dfinita = np.zeros(shape=(n,n),dtype=float)
tabla = np.concatenate((tabla,dfinita), axis=1)

# Calcula tabla, inicia en columna 3
#tama o de la matriz
[n,m] = np.shape(tabla)
diagonal = n-1
j = 3
while (j < m):
    # cada fila de columna
    i = 0
    paso = j-2 # inicia en 1
    while (i < diagonal):
        denominador = (xi[i+paso]-xi[i])
        numerador = tabla[i+1,j-1]-tabla[i,j-1]
        tabla[i,j] = numerador/denominador
        i = i+1
    diagonal = diagonal - 1
    j = j+1

# POLINOMIO con diferencias Divididas
# caso: puntos equidistantes en eje x
dDividida = tabla[0,3:]
n = len(dfinita)



x = sym.Symbol('x')
polinomio = fi[0]
for j in range(1,n,1):
    factor = dDividida[j-1]
    termino = 1
    for k in range(0,j,1):
        termino = termino*(x-xi[k])
    polinomio = polinomio + termino*factor

# simplifica multiplicando entre (x-xi)
polisimple = polinomio.expand()
```

```
# polinomio para evaluacion num rica
px = sym.lambdify(x, polisimple)

# Puntos para la gr fica
muestras = 101
a = np.min(xi)
b = np.max(xi)
pxi = np.linspace(a,b,muestras)
pfi = px(pxi)


print('polinomio con Newton: ' )
print(polisimple)
solucion = px(5000000)
print(solucion)



plt.plot(xi,fi,'o', label = 'Puntos')
plt.plot(pxi,pfi, label = 'Polinomio')
plt.legend()
plt.xlabel('xi')
plt.ylabel('fi')
plt.title('Diferencias Divididas - Newton')
plt.show()
```

El cual nos arrojo los siguientes resultados:

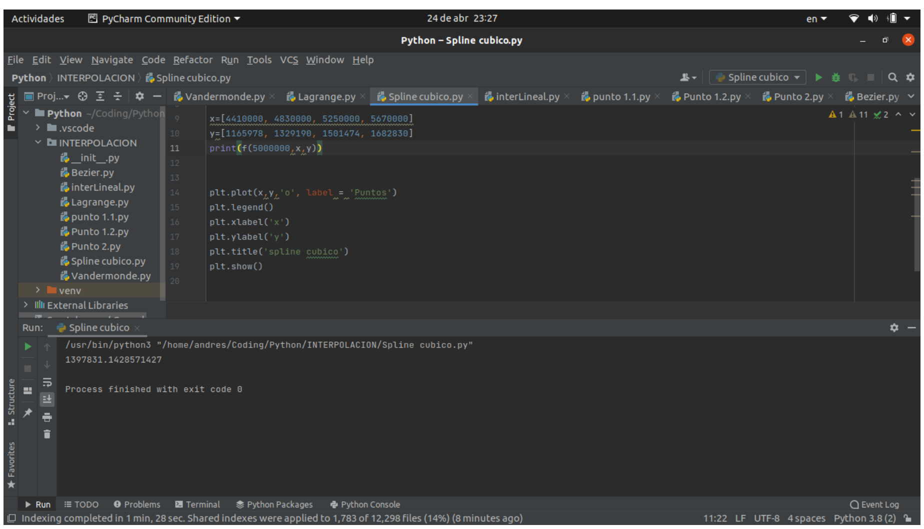


Resultado: 1397831.1428571427

METEDO SPLNE CUBICO (GRADO 3)

```
    from scipy import interpolate
import numpy as np
import matplotlib.pyplot as plt

def f(x, xp, xy):
    tck = interpolate.splrep(xp, xy)
    return interpolate.splev(x, tck)

x=[4410000, 4830000, 5250000, 5670000]
y=[1165978, 1329190, 1501474, 1682830]
print(f(5000000,x,y))


plt.plot(x,y,'o', label = 'Puntos')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.title('spline cubico')
plt.show()
```

El cual nos dio como resultado:

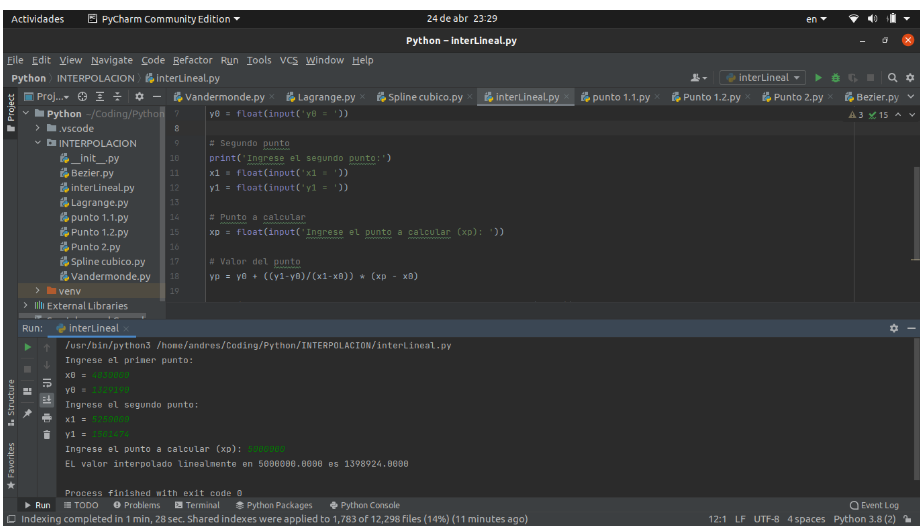


Resultado: 1397831.1428571427

METODO DE INTERPOLACION LINEAL

```
# INTERPOLACION LINEAL

# Input section
# Primer punto
print('Ingrese el primer punto:')
x0 = float(input('x0 = '))
y0 = float(input('y0 = '))

# Segundo punto
print('Ingrese el segundo punto:')
x1 = float(input('x1 = '))
y1 = float(input('y1 = '))

# Punto a calcular
xp = float(input('Ingrese el punto a calcular (xp): '))

# Valor del punto
yp = y0 + ((y1-y0)/(x1-x0)) * (xp - x0)

print('EL valor interpolado linealmente en %0.4f es %0.4f' %(xp,yp))
```

El cual nos dio los siguientes resultados:



Resultado: 1398924.0000

CONCLUSION:

Teniendo en cuenta lo anterior, pudimos determinar que el resultado evaluado en los puntos y en el punto de 5 millones coincide en los metodos de interpolacion de Nexton y en el de spline cubico.