جَامعة المَنارة
MANARA UNIVERSITY

الجلسة العملية السابعة

عنوان الجلسة: استيفاء التوابع

الغاية من الجلسة :

دراسة خوارزميات استيفاء التوابع بالطرق العددية و تحويلها الى أكواد برمجية بلغة البايثون

1-(NEWTON'S DIVIDED-DIFFERENCE INTERPOLATING)

1 – الاستيفاء بطريقة نيوتن المعتمدة على الفروق المقسومة

تحتاج هذه الطريقة الى استخراج جدول الفروق المقسومة ثم استخدامه في حساب كثيرة حدود نيوتن.

التمرين الأول:

بما أن الخوارزمية هي خوارزمية تكرارية يمكن حل التمرين باستخدام الحلقات او باستخدام التوابع العودية أو باستخدام تابع بداخله حلقة

لا حظ أن نص التمرين يطلب الحل بتابع

في بايثون :

1- استخدم تابع لايجاد جدول الفروق التقدمية بحيث يعيد قيمة معاملات كتيرة حدود استيفاء نيوتن.

2-استخدم تابع لايجاد استيفاء نيوتن من أجل قيمة 1.5

x	0	1	2	3
y	0	1	8	27

3-طبق ما سبق لايجاد استيفاء نيوتن من أجل الجدول التالي:

4- ارسم باستخدام matplotlib.pyplot منحني كثيرة الحدودالناتج مبينا كيف يمر عبر النقاط التي ترد في الجدول .

الخوارزمية:

من أجل ايجاد المعاملات:

a_list=[a0,a1,a2,a3]

يجب حساب قيم a[i] في الأعمدة المبينة في الجدول جانبا

حيث أن كل عنصر من عناصر كل عمود تحسب من العلاقة:

(y[i+1]-y[i])/(x[i+1+k]-x[i])

يمثل رقم المرتبة

-نلاحظ أن عدد العناصر يتناقص من عمود الى العمود الذي يليه

-نلاحظ أن الحساب يتوقف عندما يصبح عدد العناصر في المرتبة يساوي الواحد

اي أننا بحاجة الى حلقة خارجية تمثل المراتب و حلقة داخلية من أجل قيم عناصر x و y توجد في قلب التابع

المرتبة 0

x x[i]	y y[i]	المرتبة الأولى	المرتبة الثانية	المرتبة الثالثة
0 x[0]	0 y[0]=a[0]			
		$a[1] = F(x1,x0) = \frac{y0-y1}{x0-x1}$		
1 X[1]	1 y[1]		$a[2] = F(x2,x1,x0) = \frac{F(x2,x1)-F(x1,x0)}{x2-x1}$	
		$F(x2,x1) = \frac{y2-y1}{x2-x1}$		$a[3] = F(x3,x2,x1,x0) = \frac{F(x3,x2,x1)-F(x2,x1,x0)}{x3-x1}$
2 X[2]	8 y[2]		$F(x3,x2,x1) = \frac{F(x3,x2)-F(x2,x1)}{x3-x1}$	
		$F(x3,x2] = \frac{y3-y2}{x3-x2}$		
3 X[3]	8 y[2]			

الكود البرمجي المتعلق بالحل باستخدام التابع diff تابع حساب جدول الفروق:

```
x=[0,1,2,3]
y=[0,1,8,27]

def diff(x,y):
  a_list=[y[0]]
  for j in range(len(x)):
    Temp=[]
    for i in range(len(y)-1):
      temp=(y[i+1]-y[i])/(x[i+1+j]-x[i])
      Temp.append(temp)
    y=Temp
    a_list.append(y[0])
    if (len(y)==1):
      break
  return a_list
print(diff(x,y))
```

القائمة التي سيعيدها تابع حساب جدول الفروق

القائمة التي ستحمل عناصر المرتبة

نقل القائمة التي تحمل عناصر المرتبة الى القائمة y التي ستستخدم لحساب عناصر المرتبة التالية.

شرط ايقاف الحلقة الوصول الى عنصر وحيد في القائمة y

OUTPUT

```
[0, 1.0, 3.0, 1.0]
```

الكود البرمجي المتعلق بالحل باستخدام تابع نيوتن للاستيفاء:

```
Pn(value)=
a[0]
+
a[1]*(value-x[0])
+
a[2]*(value-x[0])* (value-x[1])
+
a[3]*(value-x[0])* (value-x[1])* (value-x[2])
+
.
.
+
a[n]* (value-x[0])* (value-x[1])* (value-x[2])*..*(value-x[n-1])
```

سنقوم بإعادة كتابة كثيرة حدود نيوتن بشكل جديد لنستنتج منه خوارزمية الحل:

ملاحظة -1- القيمة الجديدة هي ناتج ضرب القيمة القديمة بالقيمة التالية.

ملاحظة -2- دليل المعامل a يزيدعلى دليل اخر x في سلسلة المضاريب بواحد

الكود البرمجي المتعلق بالحل باستخدام مكتبة matplotlib.pyplot

```
a_value=diff(x,y)
def newton(a_value,value):
  sum=a_value[0]
  temp=1
  for i in
range(len(a_value)-1):
    temp=temp*(value-x[i])
    sum=sum+a_value[i+1]*temp
  return sum
print(newton(a_value,1.5))
```

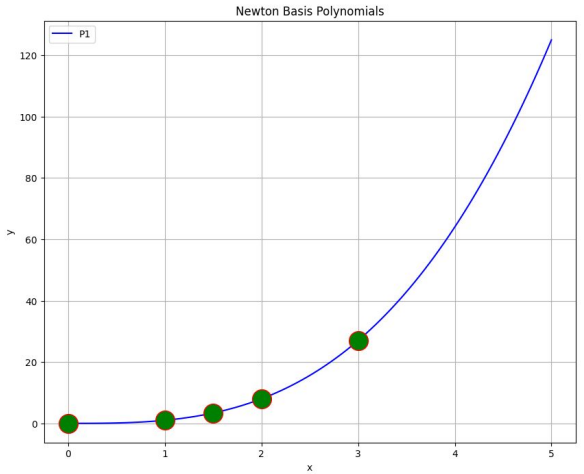
OUTPUT

```
3.375
```

الكود البرمجي المتعلق بالحل باستخدام تابع نيوتن للاستيفاء:

```
import numpy as np
import numpy.polynomial.polynomial as poly
import matplotlib.pyplot as plt

x_new = np.linspace(0, 5, 100)

fig = plt.figure(figsize = (10,8))
plt.plot(x_new, newton(a_value,x_new), 'b', label = 'P1')

f=[0,1,1.5,2,3]
for i in range(len(f)):
  plt.plot(f[i], newton(a_value,f[i]), marker="o",
markersize=20, markeredgecolor="red",
markerfacecolor="green")

plt.title('Newton Basis Polynomials')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.legend()
plt.show()
```

OUTPUT



2-(LAGRANGE INTERPOLATING POLYNOMIAL)

1- الاستيفاء بطريقة لاغرانج

لا تحتاج هذه الطريقة الى استخراج جدول الفروق المقسومة كما في حساب كثيرة حدود نيوتن.

تذكر من المحاضرة النظرية أن كثير حدود الاستيفاء بطريقة لاغرانج يعطى بالشكل التالي:

طريقة لاغرانج:

ليكن لدينا النقاط $y_0 = f(x_0), y_1 = f(x_1), \ldots\ldots, y_n = f(x_n)$

فإنّ كثير حدود الاستيفاء لاغرانج يعطى بالعلاقة:

$$f_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i) = \\ = L_0(x) f(x_0) + L_1(x) f(x_1) + \ldots\ldots L_n(x) f(x_n)$$

حيث

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

الخوارزمية المتعلقة باستيفاء لاغرانج:

```
1. Start

2. Read Data of x and y


3. Read value
4. Define function takes "value" as an argument
     4.1.Initialize list Y  ,
     4.2. For i = 0 to n-1
          4.2.1.Initialize temp to 1
          4.2.2.For j =0 to n-1
             4.2.2.1.If i ≠ j
                        temp = temp * (xp - Xj)/(Xi - Xj)
          4.2.3.Add temp to list Y
     4.3. For i = 0 to n-1
             Y = Y + temp * yi
     4.4. Return Y

5. Print y

6. Stop
```

التمرين الثاني

بما أن الخوارزمية هي خوارزمية تكرارية يمكن حل التمرين باستخدام الحلقات او باستخدام التوابع العودية أو باستخدام تابع بداخله حلقة

لاحظ أن نص التمرين يطلب الحل بشكل تابع

استخدم بايثون:

1- لكتابة تابع يعيد كثيرة حدود لاغرانج الموافقة للتابع $y=f(x)$

2-طبق ما سبق لايجاد استيفاء نيوتن من أجل ال

x	0	1	2	3
f(x)	1	2	9	28

و من أجل القيمة 1.5

3- ارسم باستخدام matplotlib.pyplot منحني كثيرة الحدود الناتج مبينا كيف يمر عبر النقاط التي ترد في الجدول .

```
import numpy as np
import numpy.polynomial.polynomial as poly
import matplotlib.pyplot as plt

x=[0,1,2,3]
y=[0,1,9,28]

def Lagrange_Interpolating(value):
  L=[]
  for i in range(len(x)):
    temp=1
    for j in range(len(x)):
      if i!=j :
        temp=temp*(value-x[j])/(x[i]-x[j])
    L.append(temp)
  Y=0
  for i in range(len(y)):
    Y=Y+L[i]*y[i]
  return Y

print(Lagrange_Interpolating(1.5))
```

جامعة المنارة
MANARA UNIVERSITY

الكود البرمجي المتعلق بالحل باستخدام التابع:

هنا استخدمنا تابع بداخله حلقات متداخلة

```
x_new = np.arange(-1.0, 3.1, 0.1)

fig = plt.figure(figsize = (10,8))
plt.plot(x_new, Lagrange_Interpolating(x_new), 'b', label = 'P1')

f=[0,1,1.5,2,3]
for i in range(len(f)):
  plt.plot(f[i], Lagrange_Interpolating(f[i]), marker= "o",
markersize=20, markeredgecolor="red", markerfacecolor="green")
plt.plot(1.5, Lagrange_Interpolating( 1.5), marker="o",
markersize=20, markeredgecolor="red", markerfacecolor="green")
plt.title('Lagrange Basis Polynomials' )
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.legend()
plt.show()
```

الكود البرمجي الخاص برسم التابع :

جَامعة المَنارة
MANARA UNIVERSITY

output