مقرر التحليل العددي

د. يمار الحموي

م.اية خيربك

م. ندى جنيدي

العملي

الفصل الثاني 2022-2023

الجلسة العملية الثالثة

عنوان الجلسة: سلاسل تايلور

الغاية من الجلسة :

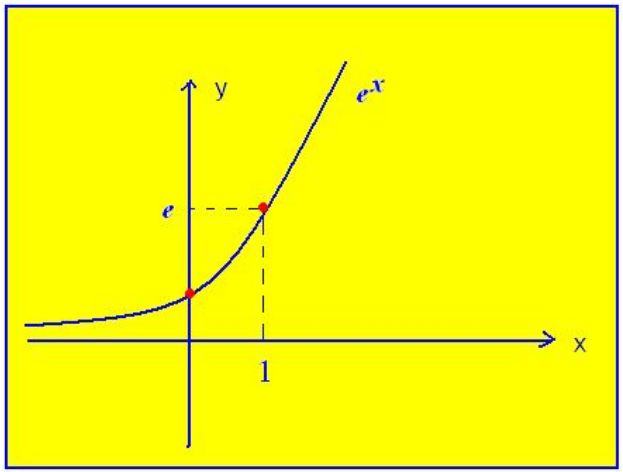
- تحويل الحل الرياضي لخوارزمية سلسلة تايلور الى برنامج بلغة البرمجة بايثون.مرة باستخدام الحلقات و مرة باستخدام التوابع.
- تطبيق مثال التابع الأسي .

في الرياضيات، مجموع تايلور أو متسلسلة تايلور هو تمثيل لدالة رياضية في شكل متسلسلة متكونة من حدود تم احتسابها باستعمال قيم اشتقاق هذه الدالة في نقطة معينة.

اخترع مفهوم متسلسلات تايلور بشكل رسمي عالم الرياضيات الأنجليزي بروك تايلور. وكان ذلك عام 1715

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f^{(3)}(x_0)}{3!}(x - x_0)^3 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n + R_n, \qquad x,\ x_0 \in (a,\ b),$$

لنأخذ المثال التالي : تعطى سلسلة تايلور للتابع الاسي بالشكل التالي:



$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + .. = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

التمرين الأول:

استخدم بايثون:

1- لحساب قيمة التابع الاسي e^x من أجل:

- قيمة x من التمط float مدخلة من قبل المستخدم .
- عدد من حدود السلسلة n و يجب أن يكون عددا صحيحا int مدخلا من قبل المستخدم.

مستخدما حلقة for من أجل تحقيق ذلك.

2- قارن القيمة الناتجة مع القيمة التي يقوم تابع exp() في مكتبة الـ math باحتسابها (أو قم بإيجاد الخطأ المطلق)

- تذكر ان: العبارة_ القيمة مدخلة من قبل المستخدم _ تعني الادخال عبر لوحة المفاتيح و بالتالي استخدام التابع input , و تذكر أن تابع الادخال input يعيد القيمة المدخلة على شكل سلسلة محرفية لذلك يجب اجراء قسر على هذا التابع وفقا للقيمة المرغوبة. فيصبح لدينا
  - قيمة x من التمط float مدخلة من قبل المستخدم .
  - x=float(input())
  - عدد من حدود السلسلة n و يجب أن يكون عددا صحيحا int مدخلا من قبل المستخدم.

بما أننا سنحتاج الى استخدام حلقة for لنبدأ بالتعرف على الشكل العام للحلقة:

```
for val in sequence:
    # statement(s)
```

- val عداد الحلقة
- sequence مجال الحلقة

Loop Over Python List

```
languages = ['Swift', 'Python', 'Go', 'JavaScript']

# access items of a list using for loop

for language in languages:

    print(language)
```

output:

```
Swift

Python

Go

JavaScript
```

Python for Loop with Python range()

```
# use of range() to define a range of values

values = range(4)

# iterate from i = 0 to i = 3

for i in values:

    print(i)
print('\n')
# you can use range(4) directly

for i in range(4):

    print(i)
```

output:

```
0

1

2

3



0

1

2

3
```

Python for loop with else

```
digits = [0, 1, 5]
for i in digits:
    print(i)
else:
    print("No items left.")
```

output:

```
0

1

5

No items left.
```

الكلمة المفتاحية `else` في حلقة `for` تحدد block من التعليمات نحتاج الى تنفيذه بعد انتهاء تنفيذ الحلقة .

Python for loop with else

```
for i in range(1, 4):
    print("The value of i = ",i)
    break
else: # Not executed as there is a break
    print("No Break")
```

يتم تنفيذ الـ *block* من التعليمات التي تلي *else* فقط اذا لم يتم انهاء حلقة *for* بواسطة التعليمة *break* .

output:

```
The value of i =  1
```

سنحتاج في البداية الى استيراد المكتبات التالية:

```
import numpy as np
import matplotlib.pyplot as plt
import math
```

```
#الجزء الأول من التمرين
x=float(input())
n=int(input())
y=0
for i in range(0,50):
    y += x**i/math.factorial(i)    #  تكافئ  y = y + x**i/math.factorial(i)
print('approximated y=',y)

#الجزء الثاني من التمرين
actual_y=math.exp(x)
print('actual_y=',actual_y)
absolute_error=abs(actual_y-y)
print('absolute_error=',absolute_error)
```

output:

```
3.3
50
approximated y= 27.11263892065789
actual_y= 27.112638920657883
absolute_error= 7.105427357601002e-15
```

⟷ يمثل هذا السهم الأحمر الازاحة Indentation و هي تعني أن هذه التعليمات متعلقة بالحلقة.

`math.factorial(i)` تابع يعيد قيمة عاملي i و هو من مكتبة الـ math

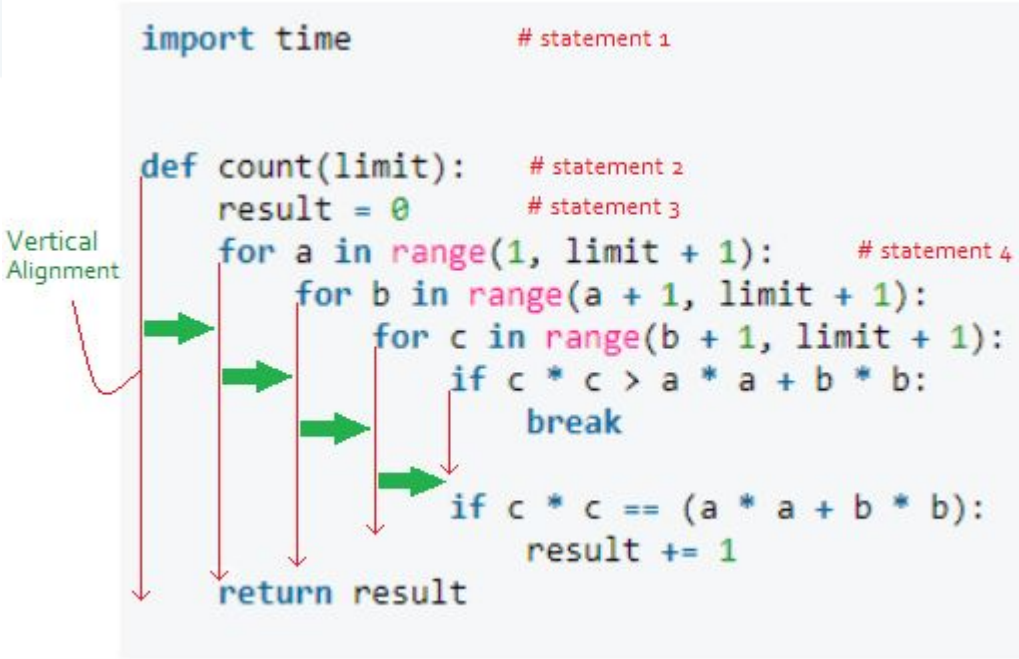
تابع abs() يعيد القيمة المطلقة للقيمة التي تمرر له.

What is the role of indentation in Python? (Why is indentation used?)

Basically, Indentation in Python is used to specify a block of code (a block of code may contain a single statement).

Or,

In other words, Indentation in Python is used to separate a statement or a block of code from the remaining blocks of code


Vertical
Alignment
import time # statement 1
def count(limit): # statement 2
result = 0 # statement 3
for a in range(1, limit + 1): # statement 4
for b in range(a + 1, limit + 1):
for c in range(b + 1, limit + 1):
if c * c > a * a + b * b:
break
if c * c == (a * a + b * b):
result += 1
return result

استخدم بايثون:

1- لحساب قيمة التابع exponential من أجل قيمة x مدخلة من قبل المستخدم و من أجل عدد من حدود السلسلة مدخل من قبل المستخدم أيضا:

- باستخدام تابع يعيد قيمة
- استخدام تابع لا يعيد قيمة.

2- قارن القيمة الناتجة مع القيمة التي يقوم تابع exp في مكتبة الـ math باحتسابها.

Types of function

There are two types of function in Python programming:

- Standard library functions - These are built-in functions in Python that are available to use.[math.exp]
- User-defined functions - We can create our own functions based on our requirements.

استخدم بايثون:

1- لحساب قيمة التابع exponential من أجل قيمة x مدخلة من قبل المستخدم و من أجل عدد من حدود السلسلة مدخل من قبل المستخدم أيضا:

- باستخدام تابع يعيد قيمة
- استخدام تابع لا يعيد قيمة.

2- قارن القيمة الناتجة مع القيمة التي يقوم تابع exp في مكتبة الـ math باحتسابها.

جَامعة المَنارة
MANARA UNIVERSITY

لنتعرف في البداية على تعريف التوابع في بايثون

Python Function Declaration

:The syntax to declare a function is

```
def function_name (arguments):
    #function body
    return
```

Here,

- `def` - keyword used to declare a function
- `function_name` - any name given to the function
- `arguments` - any value passed to function
- `return` (optional) - returns value from a function

```
#using tylor series to calculate vlaue of exponintional at
specified point

x=float(input("x= "))
n=int(input("n= "))

def Tylor_Series_Function_no_return(num_value,range_limit):
      y = 0
      for i in range(range_limit):
        y += num_value**i/math.factorial(i)
      print(y)

def Tylor_Series_Function_with_return(num_value,range_limit):
      y= 0
      for i in range(n):
        y += num_value**i/math.factorial(i)
      return y
```

```
Tylor_Series_Function_no_return(x,n)
approximated_y=Tylor_Series_Function
_with_return(x,n)
print(approximated_y)
actual_y=math.exp(x)
print(actual_y)
error=abs(actual_y-approximated_y)
print(error)
```

output:

```
x= 3.3
n= 50
27.11263892065789
27.11263892065789
27.112638920657883
7.105427357601002e-15
```