CECC421: Numerical Analysis
Lecture 2: Taylor Series

Eng. Aya Kherbek
Eng. Baher Kherbek
Faculty of Engineering
Department of Informatics
Manara University

Purpose of the Session:

*To implement **Taylor series** in Python by transforming the **mathematical solution into a program** using both loops and functions. Additionally, we will apply the method to the **exponential** function for practical demonstration.*

Introduction:

In mathematics, the **Taylor series** is a representation **of a function as an infinite sum of terms, calculated using the function's derivatives at a specific point**. It provides a powerful method for approximating complex functions using polynomials.

The concept of Taylor series was formally introduced by the English mathematician **Brook Taylor** in the year **1715**. Since then, it has become a fundamental tool in mathematical analysis, numerical methods, and computational applications.

Introduction:

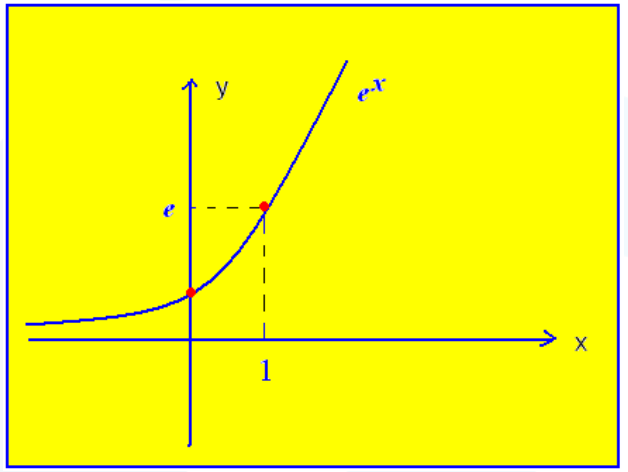
In mathematics, the **Taylor series** is a representation **of a function as an infinite sum of terms, calculated using the function's derivatives at a specific point**. It provides a powerful method for approximating complex functions using polynomials.

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

Example:

The **Taylor series** for the **exponential function** is given as follows:

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



This expansion allows us to approximate e^x using a finite number of terms, making it useful in numerical analysis and computational applications.

Example 1:

Use Python to:

1.Calculate the exponential function e^x for:

- A **float value x** entered by the user.
- An **integer n** representing the number of terms in the series, also entered by the user.

Using a **for loop** to achieve this.

2.Compare the resulting value with the one computed by the exp() function in the math library (or find the absolute error).

Remember that the phrase "*value entered by the user*" refers to input via the keyboard, which means **using the input function**. Also, the input function returns the entered value as a string, so it must be cast to the desired type.

Thus, we get:

- A float value x entered by the user:

```
x = float(input())
```

- An integer n representing the number of terms in the series, also entered by the user:

```
n = int(input())
```

For loop

```
for val in sequence:

    # statement(s)
```

- val → Loop counter (iterator variable)
- sequence → Loop range (iterable collection)

This structure iterates over each element in sequence, assigning it to val, and executing the statement(s) inside the loop.

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")
```

output:

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

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

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)
#y = y + x**i/math.factorial(i)
print('approximated y=',y)
```

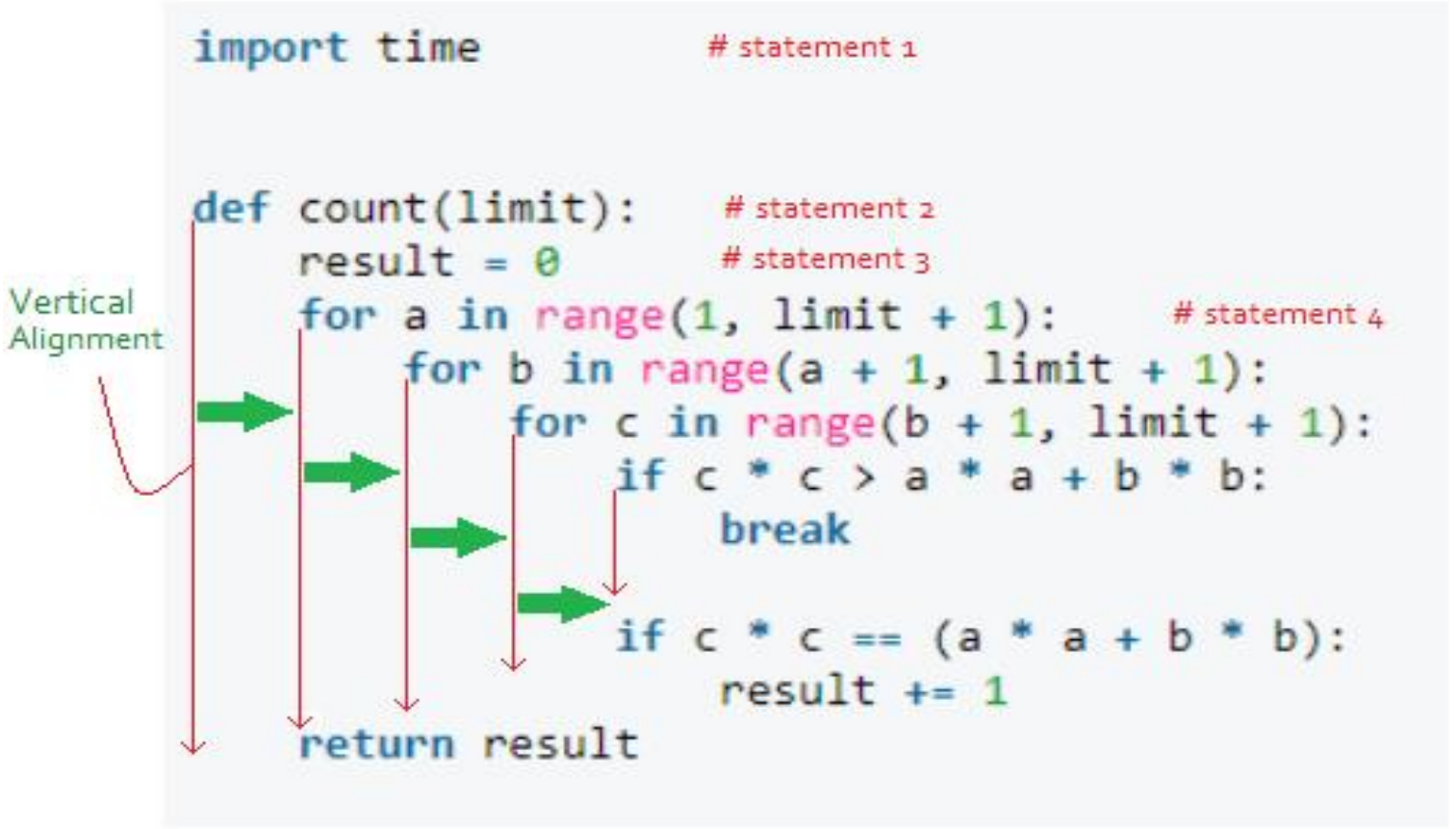
الجزء الثاني من التمرين

```
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
```

Indentation



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.

Example 2:

Use Python:

1.To calculate the exponential function for a user-inputted value of x and a user-specified number of series terms:

- Using a function that **returns** a value.
- Using a function that **does not return** a value.

2.Compare the obtained value with the value calculated using the exp function from the math library.

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 value of exponential 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(range_limit):
      y += num_value**i/math.factorial(i)
    return y
```

output:

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

```
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)
```

جَامعة
المَنارة
MANARA UNIVERSITY

Thanks for Listening