

# DASAR METODE NUMERIK

## Mengenai:

1. Pengertian metode numerik
  2. Akurasi perhitungan numerik
  3. Tahapan penyelesaian masalah dengan metode numerik
  4. Study kasus deret taylor
- 

## Pendahuluan

Persoalan yang melibatkan model matematika banyak muncul dalam berbagai disiplin ilmu pengetahuan, seperti dalam bidang fisika, kimia, ekonomi, atau pada persoalan rekayasa (*engineering*), seperti Teknik Sipil, Teknik Mesin, Elektro, dan sebagainya. Seringkali model matematika tersebut muncul dalam bentuk yang tidak ideal alias rumit. Model matematika yang rumit ini adakalanya tidak dapat diselesaikan dengan metode analitik yang sudah umum untuk mendapatkan solusi sejatinya (*exact solution*). Yang dimaksud dengan metode analitik adalah metode penyelesaian model matematika dengan rumus-rumus aljabar yang sudah baku (lazim).

Bagi rekayasawan, solusi yang diperoleh secara analitik kurang berguna untuk tujuan numerik. Persoalan rekayasa dalam prakteknya tidak selalu membutuhkan solusi dalam bentuk fungsi matematika menerus (*continuous*). Rekayasawan seringkali menginginkan solusi dalam bentuk numerik, misalnya persoalan integral tentu dan persamaan diferensial.

Para rekayasawan dan para ahli ilmu alam, dalam pekerjaannya sering berhadapan dengan persamaan matematika. Persoalan yang muncul di lapangan diformulasikan ke dalam model yang berbentuk persamaan matematika. Persamaan tersebut mungkin sangat kompleks atau jumlahnya lebih dari satu. Metode numerik, dengan bantuan komputer, memberikan cara penyelesaian persoalan matematika dengan cepat dan hasilnya dapat dibuat sedekat mungkin dengan nilai sesungguhnya.

Penggunaan komputer dalam metode numerik antara lain untuk memprogram. Langkah-langkah metode numerik diformulasikan menjadi program komputer dan ditulis dengan bahasa pemograman tertentu, seperti FORTRAN, PASCAL, C, C++, BASIC, Scilab, Python, dan sebagainya. Modul ini menggunakan pemograman Python untuk menyelesaikan permasalahan yang berhubungan dengan metode numerik. Sehingga, kita dapat membuat program pendekatan metode numerik dengan lebih mudah dan solusi persoalan selalu dapat diperoleh.

## **1.1 Pengertian Metode Numerik**

Metode numerik adalah sekumpulan metode yang menyelesaikan berbagai permasalahan matematika dengan cara memformulasikan persoalan tersebut dalam bentuk matematika sederhana agar dapat diaplikasikan ke dalam komputer dengan mudah (Hornbeck,1975). Metode numerik mampu menyelesaikan suatu sistem persamaan yang besar, persamaan yang tidak linier dan persamaan kompleks yang tidak mungkin diselesaikan secara analitis.

Berbagai masalah yang ada dalam berbagai disiplin ilmu dapat digambarkan dalam bentuk matematika dari berbagai fenomena yang berpengaruh. Misalnya gerak air dan polutan di saluran, sungai dan laut, aliran udara, perambatan panas. Biasanya fenomena yang berpengaruh tersebut cukup banyak dan sangat kompleks, dan untuk menyederhanakannya diperlukan suatu asumsi, sehingga beberapa bisa diabaikan. Meskipun telah dilakukan penyederhanaan, namun sering persamaan tersebut tidak bisa diselesaikan secara analitis. Untuk itu diperlukan metode numerik untuk menyelesaikannya.

Dalam metode numerik ini dilakukan operasi hitungan dalam jumlah yang banyak dan prosesnya berulang. Sehingga dalam prakteknya perlu bantuan komputer untuk menyelesaikan hitungan tersebut. Tanpa bantuan komputer, metode numerik tidak banyak memberikan manfaat.

Sebelum komputer berkembang pesat, penggunaan metode numerik sangatlah terbatas, hal ini karena metode numerik merupakan metode yang melakukan perhitungan yang sama secara berulang untuk mendapatkan hasil dengan galat (*error*) terkecil. Pengulangan tersebut bervariasi, mulai dari sekedar 10 iterasi hingga ribuan

iterasi untuk mendapatkan hasil dengan galat (*error*) terkecil. Kemudian komputer diciptakan, yang pada awalnya hanya untuk membantu manusia dalam berhitung, sampai berkembang menjadi salah satu kebutuhan primer manusia (Munir, 2015).

Komputer pada dasarnya diciptakan untuk membantu manusia dalam melakukan perhitungan terus menerus tanpa adanya kesalahan dan menghasilkan nilai dengan galat (*error*) terkecil dan akurat, berbeda dengan manusia yang dapat melakukan kesalahan akibat faktor kelelahan atau menurunnya fokus. Kombinasi antara metode numerik dan komputer dapat menyelesaikan perhitungan kompleks, persamaan non linear, persamaan dalam jumlah besar, geometri kompleks, dan permasalahan lainnya dengan cepat. Selain mempercepat perhitungan numerik, dengan komputer kita dapat mencoba berbagai kemungkinan solusi yang terjadi akibat perubahan beberapa parameter. Solusi yang diperoleh juga dapat ditingkatkan ketelitiannya dengan cara mengubah nilai parameter.

## 1.2 Akurasi Perhitungan Numerik

Metode numerik merupakan penyelesaian persamaan matematis secara pendekatan karena penyelesaian secara analitis (eksak) belum dapat atau sulit untuk dipecahkan serta merupakan alat yang sangat ampuh untuk menyelesaikan permasalahan dalam berbagai bidang. Penyelesaian model (persamaan) matematis dengan menggunakan metode numerik ini akan menghasilkan angka (numerik) yang bukan suatu fungsi. Deret Taylor merupakan deret yang salah satunya mendasari perhitungan numerik. Deret ini dapat digunakan untuk menghitung nilai dari suatu fungsi pada titik  $x$  apabila nilai di titik  $a$ , yang berdekatan dengan titik  $x$ , diketahui. Dengan demikian, deret Taylor dapat digunakan untuk memprediksi nilai pada suatu titik sebagai turunan dari titik yang lain. Bentuk umum dari deret Taylor dituliskan pada persamaan berikut.

$$f(x) = f(a) + f'(a)(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \frac{f'''(a)}{3!}(x - a)^3 + \dots + \frac{f^n(a)}{n!}(x - a)^n \quad (1.1)$$

Keterangan:

$f', f'', \dots, f^n$  = fungsi turunan orde satu, dua, ...,  $n$

$!$  = operator faktorial, misalnya  $3! = 3 \times 2 \times 1$

Penyelesaian secara numerik suatu persamaan matematika hanya memberikan nilai perkiraan yang mendekati nilai sebenarnya (nilai eksak), Berarti dalam penyelesaian numerik terdapat kesalahan terhadap nilai sebenarnya. Ada tiga macam kesalahan yaitu:

1. Kesalahan bawaan

Merupakan kesalahan dari nilai data. Kesalahan tersebut bisa terjadi karena kekeliruan dalam menyalin data, salah membaca skala, atau kesalahan karena kurangnya pengertian mengenai hukum-hukum fisik dari data yang diukur.

2. Kesalahan pembulatan

Kesalahan ini terjadi karena tidak diperhitungkannya beberapa angka terakhir dari suatu bilangan.

Contoh:

- 38, 24352 dibulatkan menjadi 38,24
- 106,2681 dibulatkan menjadi 106,27
- 24,65 dibulatkan menjadi 24,7

3. Kesalahan pemotongan

Hal ini terjadi karena tidak dilakukan hitungan sesuai dengan prosedur matematika yang benar. Sebagai contoh, suatu proses yang tak berhingga diganti dengan proses hingga. Seperti deret Taylor yang apabila digunakan sampai suku kedua, maka diperoleh:

$$f(x) \approx f(a) + f'(a)(x - a) + \frac{f''(a)}{2!}(x - a)^2 + E \quad (1.2)$$

Dimana E merupakan simbol kesalahan pemotongan dari deret selanjutnya. Untuk mendapatkan hasil yang lebih baik, orde yang lebih tinggi dari deret tersebut dapat digunakan.

Penyelesaian secara numerik merupakan metode pendekatan yang hasilnya diharapkan mendekati penyelesaian secara eksak (analitis). Perbedaan hasil diantara kedua hasil tersebut disebut kesalahan absolut ( $\varepsilon_a$ ), yang dapat dituliskan dalam bentuk lain, yaitu:

$$\varepsilon_a = \text{Nilai sesungguhnya (NS)} - \text{Nilai pendekatan (NP)} \quad (1.3)$$

Apabila dibandingkan dengan nilai sesungguhnya, kesalahan itu akan disebut kesalahan relatif. Kesalahan relatif tersebut dapat dibedakan menjadi dua yaitu

kesalahan relatif sesungguhnya ( $\varepsilon_{rs}$ ) dan kesalahan relatif pendekatan ( $\varepsilon_{rp}$ ). Tingkat kesalahan relatif yang kecil akan menentukan akurasi dari hasil perhitungan pendekatan.

$$\text{Kesalahan relatif sesungguhnya} : \varepsilon_{rs} = \frac{\varepsilon_a}{NS} \quad (1.4)$$

$$\text{Kesalahan relatif pendekatan} : \varepsilon_{rp} = \frac{NP_{baru} - NP_{sebelum}}{NP_{baru}} \quad (1.5)$$

**Contoh soal:**

Pengukuran tinggi gedung dan pohon menghasilkan 2220 cm dan 442 cm. Apabila panjang yang benar (eksak) adalah 2222 cm dan 444 cm.

Hitunglah: a. kesalahan absolut

b. kesalahan relatif

Jawab:

a. Kesalahan absolut

Gedung:

$$\varepsilon_a = 2222 - 2220$$

$$= 2 \text{ cm}$$

Pohon:

$$\varepsilon_a = 444 - 442$$

$$= 2 \text{ cm}$$

b. Kesalahan relatif

Gedung:

$$\varepsilon_{rs} = \frac{\varepsilon_a}{NS} = \frac{2}{2222} \times 100\% = 0,09\%$$

Pohon:

$$\varepsilon_{rs} = \frac{\varepsilon_a}{NS} = \frac{2}{444} \times 100\% = 0,45\%$$

Contoh tersebut memperlihatkan bahwa pengukuran tinggi gedung dan pohon memiliki kesalahan absolut yang sama yaitu 2 cm tetapi kesalahan relatif pohon lebih besar. Artinya bahwa pengukuran tinggi gedung memberikan hasil yang baik (memuaskan) sedangkan pengukuran pohon tidak memuaskan.

### 1.3 Tahapan Penyelesaian Masalah dengan Metode Numerik

#### a. Pembentukan model matematika

Permasalahan diformulasikan ke dalam model-model persamaan matematika yang dapat menggambarkan permasalahan tersebut secara menyeluruh.

#### b. Penyederhanaan model

Melakukan penyederhanaan model persamaan matematika yang sudah di peroleh sebelumnya, penyederhanaan ini bertujuan untuk mempermudah dalam mencari solusi permasalahan.

#### c. Formulasi numerik

Setelah model matematika yang sederhana diperoleh selanjutnya adalah memformulasikannya secara numerik:

##### i) Menentukan metode yang dipakai

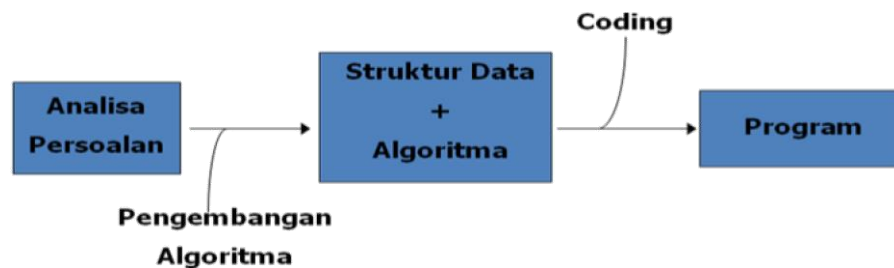
Pemilihan metode didasari pada pertimbangan:

1. apakah metode tersebut teliti?
2. apakah metode tersebut mudah diprogram dan waktu pelaksanaannya cepat?
3. apakah metode tersebut tidak peka terhadap perubahan data yang cukup kecil?

##### ii) Menyusun algoritma dari metode numerik yang dipakai

#### d. Pemrograman

Membuat program dengan menggunakan bahasa tertentu sesuai dengan algoritma yang sudah dibuat pada tahap formulasi numerik. Berikut adalah bentuk dari pemrograman terstruktur.



Gambar 1: bentuk pemrograman terstruktur

Keterangan:

- Analisa Persoalan

Persoalan apa yang perlu diselesaikan (tujuan atau program).

Bagaimana program harus berbuat untuk menyelesaikan persoalan

- Struktur data dan algoritma

secara garis besar yaitu:

1. Rancang struktur data yang digunakan, sedemikian rupa sehingga mudah merancang algoritmanya dan memberikan proses yang efisien.
2. Dengan memahami masalah yang dimiliki, rancang setiap langkah yang diperlukan untuk menyelesaikan (bisa dalam bentuk *flowchart* ataupun *pseudocode* dan pilih teknik-teknik yang efisien (*design algorithm*).
3. Mencoba algoritma yang dibuat dengan data sederhana.

Di dalam menyusun algoritma, hanya dikenal 3 jenis struktur proses :

1. Struktur sederhana ( <i>sequence</i> )	: Perintah-perintah dilakukan secara berurutan.
2. Struktur berulang ( <i>repetition/loop</i> )	: Satu/sekumpulan perintah dilakukan berulang-ulang.
3. Struktur bersyarat	: Satu/sekumpulan perintah dilakukan jika suatu kondisi dipenuhi.

Algoritma yang disusun berdasarkan struktur di atas disebut algoritma terstruktur, sedangkan program yang dibuat berdasarkan algoritma terstruktur dikatakan sebagai program terstruktur.

Alasan menggunakan program terstruktur :

- Jika kita telah terbiasa menganalisa dan menyusun program untuk suatu masalah dengan menggunakan teknik yang sama, maka pemecahan masalah (analisa, desain, penyusunan program) akan menjadi lebih cepat dan mengurangi galat (*error*).
- Jika persoalan dilakukan oleh suatu *teamwork*, dan semua *programmer* menggunakan teknik yang sama, maka akan lebih mudah bagi seorang *programmer* untuk dapat membaca pekerjaan *programmer* yang lain dan mudah dalam pengembangan atau perbaikan program nantinya.

- Program

Tahapan program INPUT → PROSES → OUTPUT.

- Coding : susun pengkodean-nya sesuai dengan algoritma dan struktur data yang telah dibuat beserta Keteranganannya.
- Debugging : memperbaiki kesalahan sintaks (tahap compiling), kesalahan logika dan kesalahan run-time (tahap eksekusi).
- Testing : menguji dengan data yang telah diketahui hasil outputnya.
- Dokumentasi.

#### e. Operasional

Program yang sudah dihasilkan diujicoba untuk menyelesaikan permasalahan tersebut dan permasalahan lainnya yang serupa kemudian dicatat hasilnya.

#### f. Evaluasi

Hasil program dievaluasi, untuk mengetahui apakah pembentukan model, metode numerik yang digunakan, dan program sudah tepat atau belum.

### 1.4 Study Kasus Deret Taylor

Deret Taylor merupakan salah satu alat yang sangat penting dalam metode numerik untuk menurunkan suatu metode numerik. Deret Taylor digunakan untuk menentukan nilai hampiran dari fungsi dalam bentuk polinom yang merupakan deret geometri tak hingga sehingga menghasilkan nilai hampiran yang berbeda dengan nilai sejatinya. Perbedaan tersebut adalah galat (*error*). Teorema Taylor merupakan teorema yang menyatakan bahwa nilai suatu fungsi bisa didapat tidak hanya dengan memasukkan nilai ke dalam fungsi tertentu secara langsung, tetapi juga bisa dihampiri dengan menggunakan polinomial (Hazewinkel, Michiel, 2001).

Sebelum menyelesaikan persoalan deret Taylor, berikut beberapa penjabaran mengenai rumus-rumus dasar dari turunan (*derivative*).

1.  $f(x) = x^n$  maka  $f'(x) = nx^{n-1}$
2.  $f(x)$  suatu fungsi konstanta maka  $f'(x) = 0$
3.  $f(x)$  suatu fungsi trigonometri:
  - a.  $f(x) = \sin x$  maka  $f'(x) = \cos x$



- b.  $f(x) = \cos x$  maka  $f'(x) = -\sin x$
  - c.  $f(x) = \tan x$  maka  $f'(x) = \sec^2 x$
  - d.  $f(x) = \cotan x$  maka  $f'(x) = -\operatorname{cosec}^2 x$
  - e.  $f(x) = \sec x$  maka  $f'(x) = \sec x \tan x$
  - f.  $f(x) = \operatorname{cosec} x$  maka  $f'(x) = -\operatorname{cosec} x \cotan x$
4.  $f(x)$  suatu fungsi eksponen:
- a.  $f(x) = a^x$  maka  $f'(x) = a^x \ln a$
  - b.  $f(x) = e^x$  maka  $f'(x) = e^x$

**Contoh soal:**

Carilah nilai  $f(1)$  untuk  $f(x) = e^{x+3}$  untuk  $x_0 = 0$  hingga 5 iterasi.

**Jawab:**

- **Dengan cara manual**

- Perhitungan normal

$$f(x) = e^{x+3}$$

$$f(1) = e^{1+3} = 54,59815$$

Nilai 54,59815 merupakan hasil yang benar atau disebut dengan solusi sejati. Kemudian untuk perbandingan, kita gunakan deret Taylor untuk mendekati nilai tersebut.

- Perhitungan Taylor (Nilai mendekati dengan 5 iterasi)

**Iterasi 1**

$$f(x) = (x_0)$$

$$f(1) = e^{x_0+3} = e^{0+3} = 20,085537$$

**iterasi 2**

$$f(x) = (x_0) + f'(x_0)(x - x_0)$$

$$f(1) = 20,085537 + f'(x_0)(x - x_0)$$

$$f(1) = 20,085537 + (e^{0+3})(x - x_0)$$

$$f(1) = 20,085537 + (e^{0+3})(1 - 0)$$

$$f(1) = 20,085537 + 20,085537 = 40,171074$$

**Iterasi 3**

$$f(x) = (x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2$$

$$f(1) = 40,171074 + \frac{f''(x_0)}{2!}(x - x_0)^2$$

$$f(1) = 40,171074 + \frac{e^{x_0+3}}{2!}(x - x_0)^2$$

$$f(1) = 40,171074 + \frac{e^{0+3}}{2!}(1 - 0)^2$$

$$f(1) = 40,171074 + 10,042768 = 50,213842$$

**Iterasi 4**

$$f(x) = (x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f'''(x_0)}{3!}(x - x_0)^3$$

$$f(1) = 50,213842 + \frac{f'''(x_0)}{3!}(x - x_0)^3$$

$$f(1) = 50,213842 + \frac{e^{x_0+3}}{3!}(x - x_0)^3$$

$$f(1) = 50,213842 + \frac{e^{0+3}}{3!}(1 - 0)^3$$

$$f(1) = 50,213842 + 3,3475895 = 53,561431$$

**Iterasi 5**

$$f(x) = f(a) + f'(a)(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \frac{f'''(a)}{3!}(x - a)^3 + \frac{f''''(a)}{4!}(x - a)^4$$

$$f(1) = 53,561431 + \frac{f''''(x_0)}{4!}(x - x_0)^4$$

$$f(1) = 53,561431 + \frac{e^{x_0+3}}{4!}(x - x_0)^4$$

$$f(1) = 53,561431 + \frac{e^{0+3}}{4!}(1 - 0)^4$$

$$f(1) = 53,561431 + 0,8368974 = 54,398329$$

Nilai yang di dapat dari 5 buah iterasi adalah 54,398329, nilai ini merupakan nilai pendekatan untuk perhitungan normal yang dilakukan sebelumnya, sehingga untuk mendapatkan nilai galat (*error*) atau nilai sisa yang dihasilkan dari perhitungan menggunakan deret Taylor adalah  $R_n(x)$ , untuk nilai galat (*error*) dari iterasi tersebut dapat dihitung sebagai berikut:

$$f(1) = 54,398329 + E$$

$$54,59815 = 54,398329 + E$$

$$E = 54,398329 - 54,398329$$

$$E = 0,199821$$

Nilai E akan terus mengecil jika iterasi yang dilakukan semakin banyak, dan nilai pendekatan pun semakin akurat jika nilai galat (*error*) semakin kecil. Karena banyaknya iterasi yang perlu dilakukan untuk mendapatkan galat maka dibuatlah sebuah program untuk mempermudah proses perhitungan dan menghindari kesalahan yang umum dalam perhitungan manual. Program tersebut dibuat di dalam aplikasi Python.

- **Dengan Menggunakan Python**

a. Deret Taylor

$$f(x) = f(a) + f'(a)(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \frac{f'''(a)}{3!}(x - a)^3 + \dots + \frac{f^n(a)}{n!}(x - a)^n$$

Masukkan  $f(x) = e^{x+3}$  dan  $x_0 = 0$  kedalam persamaan tersebut

$$e^{x+3} = e^{0+3} + e^{0+3}(x - 0) + \frac{e^{0+3}}{2!}(x - 0)^2 + \frac{e^{0+3}}{3!}(x - 0)^3 + \dots$$

b. Sederhanakan agar deret dapat diubah kedalam algoritma dan program dengan lebih mudah

$$e^{x+3} = e^3 + \frac{e^3x}{1!} + \frac{e^3x^2}{2!} + \frac{e^3x^3}{3!} + \dots$$

c. Mulai bentuk algoritma untuk memenuhi persamaan di atas

d. Tuangkan algoritma tersebut kedalam bentuk program.

- Untuk membuat program dari bentuk yang sudah disederhanakan hal yang harus dilakukan adalah membuka aplikasi Idle Python maka aplikasi akan membuka jendela *console* Python lalu pilih menu File lalu pilih New File maka aplikasi akan membuka jendela *nodus script* untuk memuat kode program.



*Gambar 3: Jendela console Python dan Nodus Script Python*

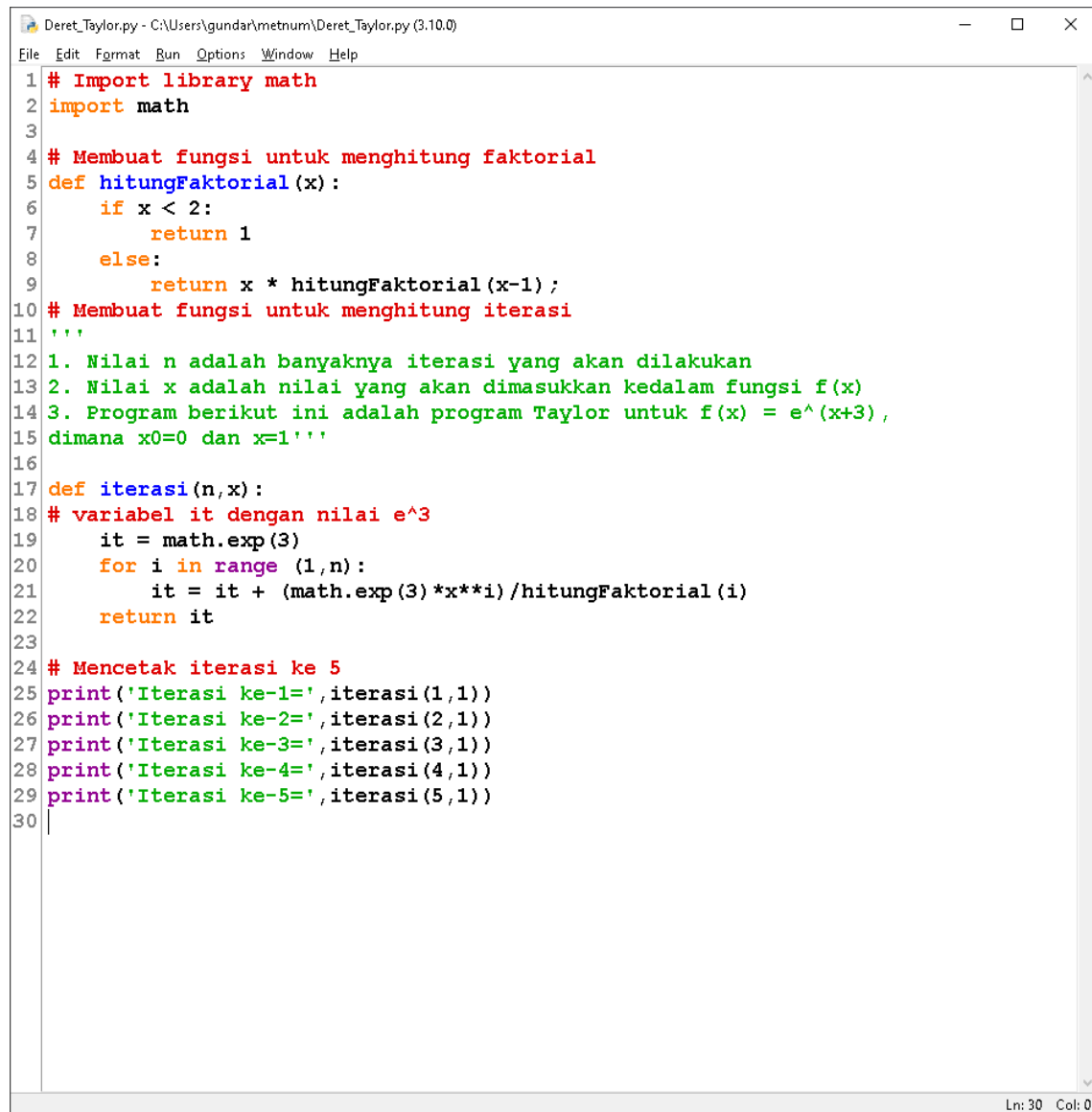
- Langkah selanjutnya yaitu menuliskan program pada lembar kerja *script* Python seperti di bawah ini, lalu simpan program dengan nama `Deret_Taylor.py`, selanjutnya lakukan *execute* program dengan klik menu Run atau dengan F5 pada *keyboard*.

```
# Import library math
import math

# Membuat fungsi untuk menghitung faktorial
def hitungFaktorial(x):
    if x < 2:
        return 1
    else:
        return x * hitungFaktorial(x-1);
# Membuat fungsi untuk menghitung iterasi
'''
1. Nilai n adalah banyaknya iterasi yang akan dilakukan
2. Nilai x adalah nilai yang akan dimasukkan kedalam
fungsi f(x)
3. Program berikut ini adalah program Taylor untuk  $f(x) = e^{(x+3)}$ ,
dimana  $x_0=0$  dan  $x=1$ '''

def iterasi(n,x):
    # variabel it dengan nilai  $e^3$ 
    it = math.exp(3)
    for i in range (1,n):
        it = it + (math.exp(3)*x**i)/hitungFaktorial(i)
    return it

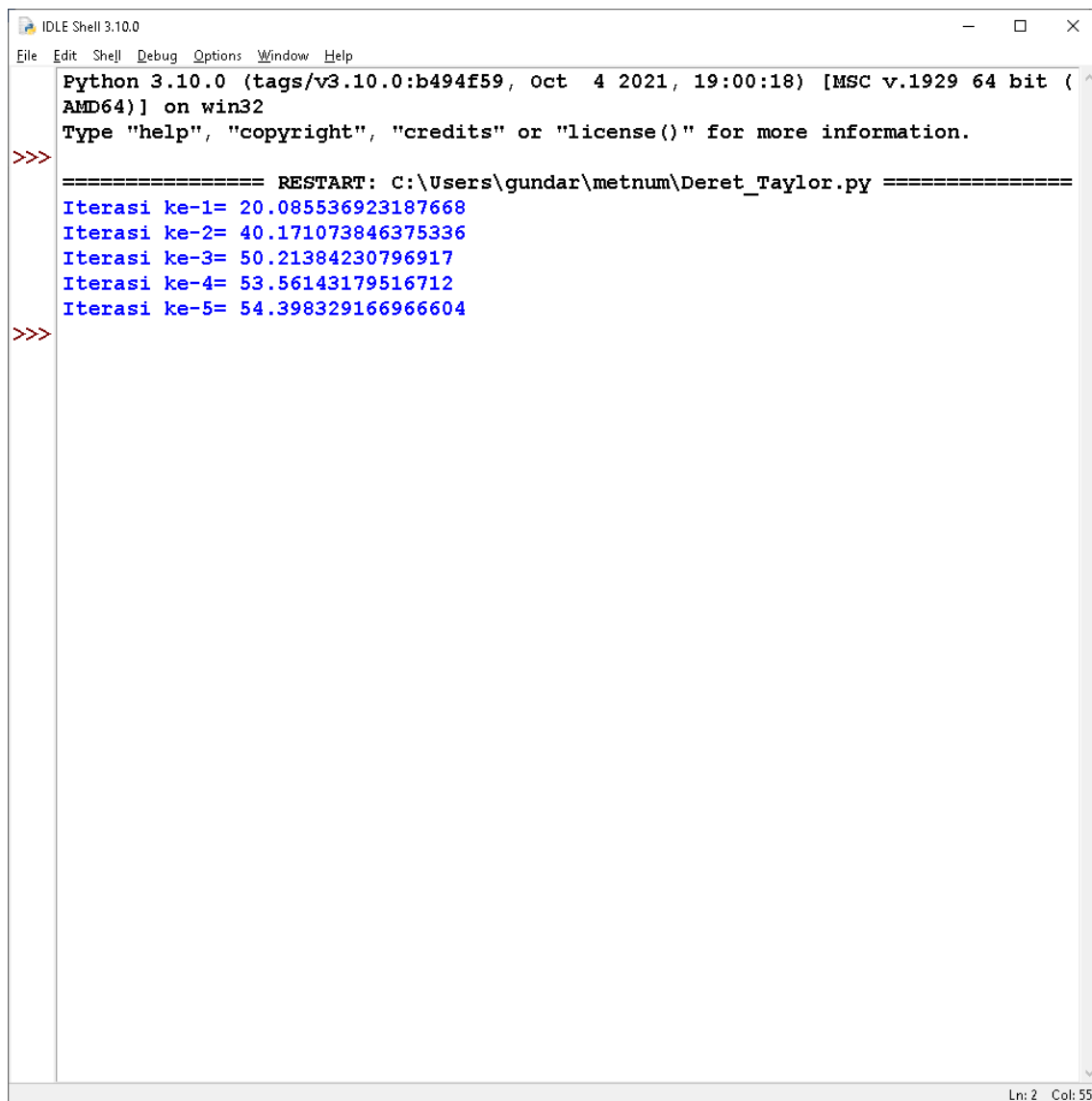
# Mencetak iterasi ke 5
print('Iterasi ke-1=',iterasi(1,1))
print('Iterasi ke-2=',iterasi(2,1))
print('Iterasi ke-3=',iterasi(3,1))
print('Iterasi ke-4=',iterasi(4,1))
print('Iterasi ke-5=',iterasi(5,1))
```



```
Deret_Taylor.py - C:\Users\gundar\metnum\Deret_Taylor.py (3.10.0)
File Edit Format Run Options Window Help
1 # Import library math
2 import math
3
4 # Membuat fungsi untuk menghitung faktorial
5 def hitungFaktorial(x):
6     if x < 2:
7         return 1
8     else:
9         return x * hitungFaktorial(x-1);
10 # Membuat fungsi untuk menghitung iterasi
11 '''
12 1. Nilai n adalah banyaknya iterasi yang akan dilakukan
13 2. Nilai x adalah nilai yang akan dimasukkan kedalam fungsi f(x)
14 3. Program berikut ini adalah program Taylor untuk f(x) = e^(x+3),
15 dimana x0=0 dan x=1'''
16
17 def iterasi(n,x):
18 # variabel it dengan nilai e^3
19     it = math.exp(3)
20     for i in range (1,n):
21         it = it + (math.exp(3)*x**i)/hitungFaktorial(i)
22     return it
23
24 # Mencetak iterasi ke 5
25 print('Iterasi ke-1=',iterasi(1,1))
26 print('Iterasi ke-2=',iterasi(2,1))
27 print('Iterasi ke-3=',iterasi(3,1))
28 print('Iterasi ke-4=',iterasi(4,1))
29 print('Iterasi ke-5=',iterasi(5,1))
30 |
```

Ln: 30 Col: 0

Gambar 4: Deret\_Taylor.py

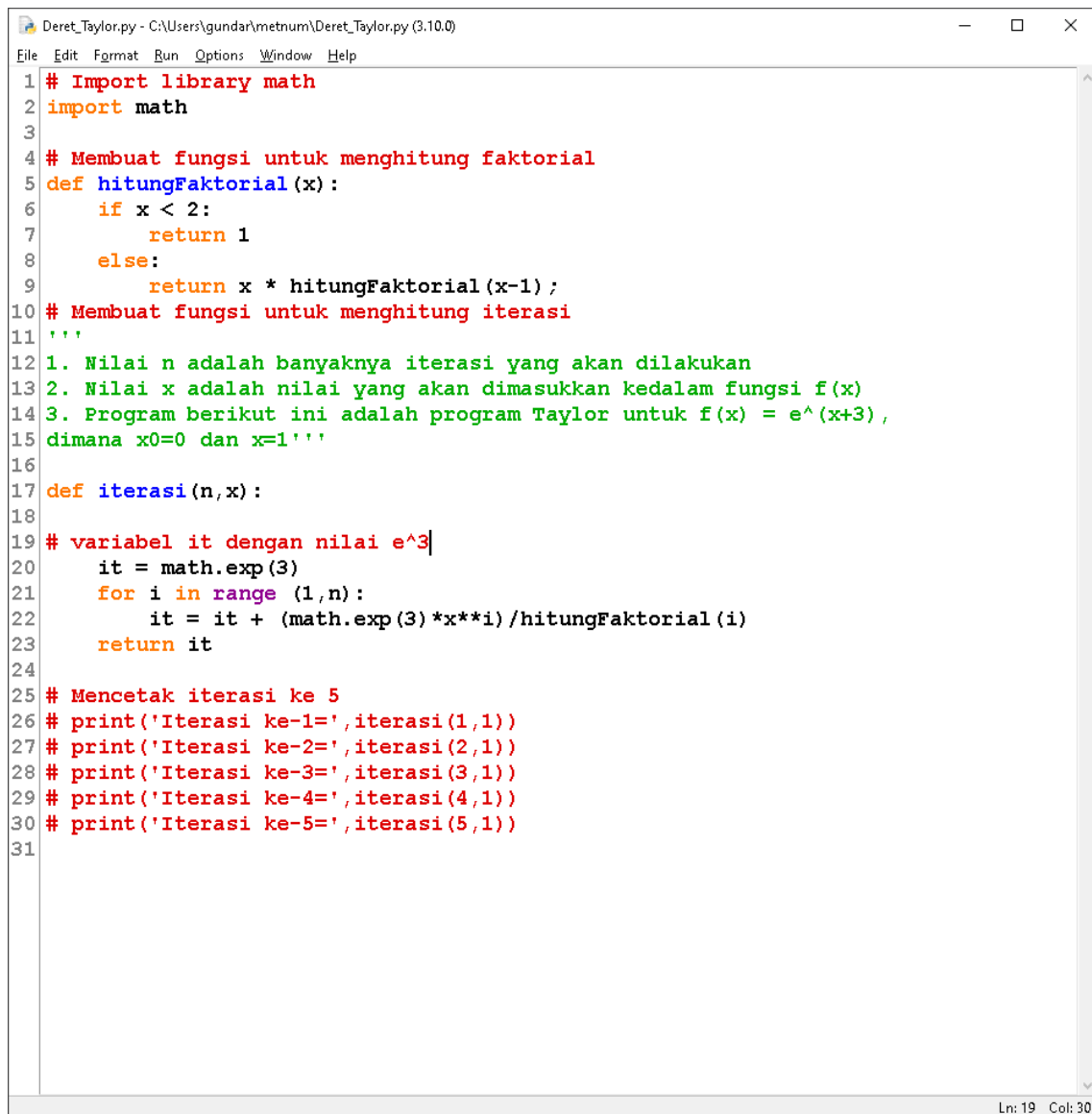


```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\gundar\metnum\Deret_Taylor.py =====
Iterasi ke-1= 20.085536923187668
Iterasi ke-2= 40.171073846375336
Iterasi ke-3= 50.21384230796917
Iterasi ke-4= 53.56143179516712
Iterasi ke-5= 54.398329166966604
>>>
```

*Gambar 5: Hasil dari menjalankan file program Deret\_Taylor.py*

Selain dengan cara memanggil langsung fungsi iterasi pada file program, dapat juga dilakukan dengan memanggil fungsi iterasi pada *console* yang diikuti dengan banyaknya iterasi yang dilakukan ( $n$ ) dan nilai yang akan dimasukkan kedalam fungsi  $f(x)$ , dapat dilihat pada gambar di bawah ini.

- Buat komentar untuk perintah print, lalu simpan program dan jalankan program dengan klik Run atau F5 pada *keyboard*.



```
Deret_Taylor.py - C:\Users\gundar\metnum\Deret_Taylor.py (3.10.0)
File Edit Format Run Options Window Help
1 # Import library math
2 import math
3
4 # Membuat fungsi untuk menghitung faktorial
5 def hitungFaktorial(x):
6     if x < 2:
7         return 1
8     else:
9         return x * hitungFaktorial(x-1);
10 # Membuat fungsi untuk menghitung iterasi
11 '''
12 1. Nilai n adalah banyaknya iterasi yang akan dilakukan
13 2. Nilai x adalah nilai yang akan dimasukkan kedalam fungsi f(x)
14 3. Program berikut ini adalah program Taylor untuk f(x) = e^(x+3),
15 dimana x0=0 dan x=1'''
16
17 def iterasi(n,x):
18
19 # variabel it dengan nilai e^3|
20     it = math.exp(3)
21     for i in range (1,n):
22         it = it + (math.exp(3)*x**i)/hitungFaktorial(i)
23     return it
24
25 # Mencetak iterasi ke 5
26 # print('Iterasi ke-1=',iterasi(1,1))
27 # print('Iterasi ke-2=',iterasi(2,1))
28 # print('Iterasi ke-3=',iterasi(3,1))
29 # print('Iterasi ke-4=',iterasi(4,1))
30 # print('Iterasi ke-5=',iterasi(5,1))
31
Ln: 19 Col: 30
```

Gambar 6: Hasil dari menjalankan file program Deret\_Taylor.py



- Pada jendela *console* panggil fungsi iterasi yang sebelumnya sudah dibuat pada file Deret\_Taylor.py dengan mengetikan `iterasi(n, x)` dimana `n` adalah banyaknya iterasi yang dilakukan dan `x` adalah nilai `x` yang diketahui. Lalu tekan Enter pada *keyboard*.



```
"IDLE Shell 3.10.0"
File Edit Shell Debug Options Window Help
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\gundar\metnum\Deret_Taylor.py =====
Iterasi ke-1= 20.085536923187668
Iterasi ke-2= 40.171073846375336
Iterasi ke-3= 50.21384230796917
Iterasi ke-4= 53.56143179516712
Iterasi ke-5= 54.398329166966604
>>>
===== RESTART: C:\Users\gundar\metnum\Deret_Taylor.py =====
>>> iterasi(5, 1
      (n, x)
```

Gambar 7: Console python

- Maka hasil dari mencari deret taylor untuk nilai  $f(1)$  dimana diketahui  $f(x) = e^{x+3}$ ,  $x_0 = 0$ , pada iterasi ke 5 adalah 54.398329166966604. Seperti gambar di bawah ini



```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\gundar\metnum\Deret_Taylor.py =====
Iterasi ke-1= 20.085536923187668
Iterasi ke-2= 40.171073846375336
Iterasi ke-3= 50.21384230796917
Iterasi ke-4= 53.56143179516712
Iterasi ke-5= 54.398329166966604
>>>
===== RESTART: C:\Users\gundar\metnum\Deret_Taylor.py =====
>>> iterasi(5, 1)
54.398329166966604
>>>
```

Gambar 8: Hasil dari iterasi(5, 1)

## Referensi

- Hazewinkel, Michiel ed. (2001). "Kolmogorov-Smirnov test", Encyclopedia of Mathematics. Springer, ISBN 978-1-55608-010-4.
- Munir. (2015). *Multimedia Konsep dan Aplikasi dalam Pendidikan*. Bandung: Alfabeta
- Sanjaya WS, M. (2015). *Metode Numerik Berbasis Python*. Yogyakarta: Penerbit Gava Media.
- Sholiun, & Fatomi, Z. S. (2021). *Pemrograman dan Komputasi Numerik*. Yogyakarta: Gadjah Mada University Press.
- Triatmodjo, B. (2018). *Metode Numerik*. Yogyakarta: Beta Offset