



---

# Laporan Praktikum Algoritma & Pemrograman

Semester Genap 2024/2025

SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.

SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.

NIM	<71231046>
Nama Lengkap	<FREIRE HANAN PUTRA>
Minggu ke / Materi	14 / REKURSIF

PROGRAM STUDI INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI  
UNIVERSITAS KRISTEN DUTA WACANA  
YOGYAKARTA  
2025

## **BAGIAN 1: MATERI MINGGU INI (40%)**

### **MATERI 1 Pengertian Rekursif**

Fungsi rekrusif merupakan fungsi yang berisi oleh dirinya sendiri atau bisa juga fungsi yang mendeklarasikan dirinya sendiri. Fungsi rekrusif juga sering disebut dengan fungsi yang memanggil dirinya sendiri. Fungsi reksusif adalah fungsi matematis yang berulang dan memiliki pola yang terstruktur, namun biasanya fungsi ini perlu untuk diperhatikan karena fungsi ini dapat berhenti dan agar tidak menghabiskan memori. Fungsi rekrusif adalah fungsi yang harus digunakan dengan perlu perhitungan karena fungsi rekrusif dapat bersifat unlimited loop dan akan menyebabkan program hang up.

Fungsi ini akan terus berjalan sampai kondisi terpenuhi.

Maka dari itu dalam sebuah fungsi rekursif perlu ada 2 blok penting antara lain yang menjadi titik berhenti dari sebuah proses rekursif dan blok yang memanggil dirinya sendiri. Ada dalam 2 bagian dalam fungsi rekursif yaitu antara lain:

- Basecase merupakan bagian Dimana penentu dari fungsi rekursif itu sendiri
- Rekursif case merupakan bagian Dimana terdapat statemen yang akan terus diulang terus menerus hingga mencapai basecase.

### **MATERI 2 Kelebihan dan Kekurangan**

Ada beberapa keunggulan dari fungsi rekursif, yaitu sebagai berikut:

- Kode program yang disusun lebih singkat dan elegan
- Masalah kompleks yang ada di fungsi rekursif dapat di breakdown dan analisis menjadi sub masalah yang lebih kecil di dalam sebuah fungsi rekursif.

Disisi lain ada beberapa kekurangan dari fungsi rekursif, yaitu sebagai berikut:

- Memakan memori yang lebih besar karena setiap kali bagian dirinya dipanggil maka dibutuhkan sejumlah runag memori tambahan
- Mengorbanan efesiensi dan kecepatan dalam mengolah data
- Fungsi rekursif sulit dilakukan debugging dan kadang sulit dimengerti.

### **MATERI 3 Bentuk Umum dan Studi Kasus**

Adapun bentuk umum dari fungsi rekursif pada python sebagai berikut:

```
1     def function_name(parameter_list):  
2         ...  
3         function_name(...)  
4         ...
```

Pada dasarnya semua fungsi rekursif pasti memiliki Solusi yang interatifnya. Misal pada contoh kasus factorial berikut:

Factorial adalah menghitung perkalian deret angka  $1 \times 2 \times 3 \times \dots \times n$ .

Algoritma untuk menghitung factorial adalah dapat dilakukan seperti pada Langkah dibawah ini:

- Tanyakan  $n$
- Siapkan variabel total untuk menampung hasil perkalian factorial dan set nilai awal dengan 0
- Loop dari  $i=1$  hingga  $n$  untuk mengerjakan:
- $\text{Total} = \text{total} * i$
- Setelah itu print

Dengan menggunakan rekursif tersebut maka factorial dapat dihitung dengan rumus pada gambar dibawah ini:

$$\text{fact}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot \text{fact}(n - 1) & \text{if } n > 0 \end{cases}$$

Berdasarkan rumus diatas dapat dibuat pseudocode secara rekursif yang dapat dibuat adalah:

## Pseudocode (recursive):

```
function factorial is:
input: integer  $n$  such that  $n \geq 0$ 
output:  $[n \times (n-1) \times (n-2) \times \dots \times 1]$ 

    1. if  $n$  is 0, return 1
    2. otherwise, return  $[n \times \text{factorial}(n-1)]$ 

end factorial
```

### Tujuan Fungsi

- Fungsi ini dirancang untuk menghitung nilai faktorial dari suatu bilangan bulat yang tidak negatif.
- Nilai faktorial dilambangkan sebagai  $n!$ , yang merupakan hasil perkalian berurutan dari  $n$  hingga 1, misalnya  $n! = n \times (n-1) \times (n-2) \times \dots \times 1$ .

### Input

- Parameter yang diberikan kepada fungsi adalah sebuah angka bulat  $n$ .
- Nilai  $n$  harus memenuhi syarat lebih besar atau sama dengan nol ( $n \geq 0$ ), karena faktorial tidak didefinisikan untuk bilangan negatif.

### Output

- Fungsi akan menghasilkan nilai faktorial dari  $n$ , yaitu hasil akhir dari proses perkalian berurutan tersebut.

### Logika Dasar Rekursi

1. Kasus Dasar (Base Case):
  - Saat nilai  $n$  adalah 0, fungsi akan langsung mengembalikan nilai 1.
  - Hal ini sesuai dengan aturan matematika bahwa  $0! = 1$ , dan menjadi titik henti dalam proses rekursi.

## 2. Kasus Rekursif (Recursive Case):

- Untuk nilai  $n$  yang lebih besar dari 0, fungsi akan menghitung nilai  $n \times \text{factorial}(n - 1)$ .
- Dalam langkah ini, fungsi memanggil dirinya sendiri dengan nilai satu angka lebih kecil, menciptakan rantai perhitungan sampai mencapai base case.

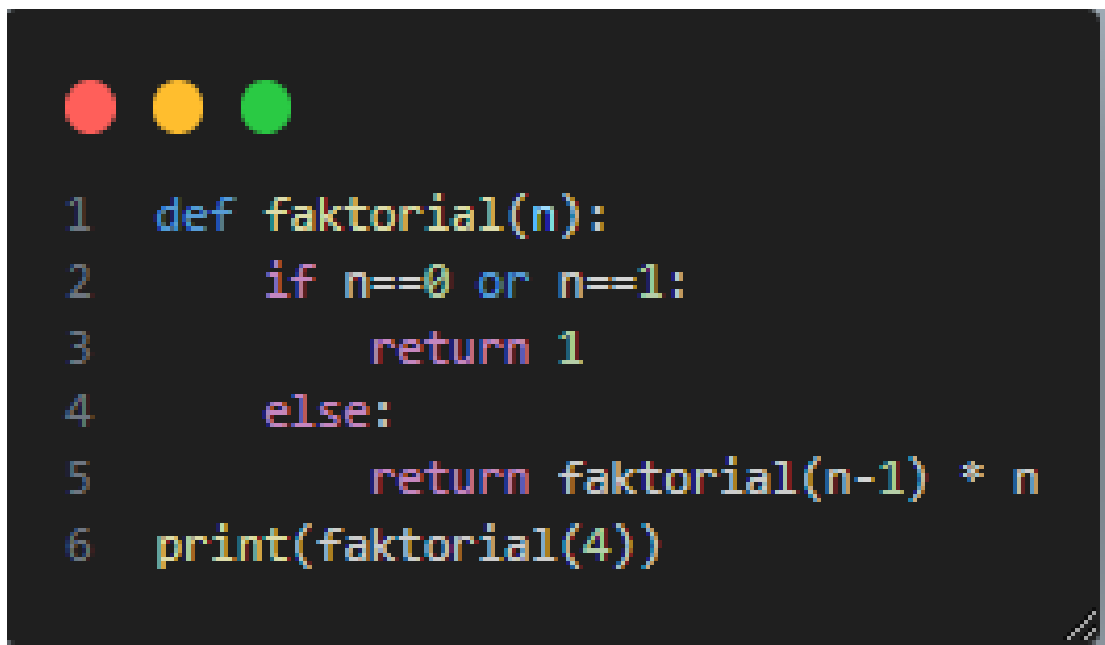
### Contoh Alur Eksekusi

Jika kita memanggil `factorial(3)`, maka proses yang terjadi adalah sebagai berikut:

- `factorial(3)` memanggil  $3 \times \text{factorial}(2)$
- `factorial(2)` memanggil  $2 \times \text{factorial}(1)$
- `factorial(1)` memanggil  $1 \times \text{factorial}(0)$
- `factorial(0)` mengembalikan 1 (karena base case)
- Proses kemudian kembali naik:  $1 \times 1 \times 2 \times 3 = 6$

### Keunggulan Menggunakan Rekursi

- Struktur kode menjadi lebih ringkas dan mudah dipahami, karena mengikuti pola pemecahan masalah secara alami.
- Sangat cocok digunakan untuk permasalahan yang bersifat berulang atau dapat dibagi menjadi submasalah serupa, seperti perhitungan faktorial, deret Fibonacci, dan lain-lain.



```
1 def faktorial(n):
2     if n==0 or n==1:
3         return 1
4     else:
5         return faktorial(n-1) * n
6 print(faktorial(4))
```

Penjelasan:

- Jadi basecasenya terdapat pada pemisalan  $n==0$  or  $n==1$ :
- Jika sesuai dengan definisi factorial dalam matematika, yaitu  $0! = 1$  dan factorial  $1! = 1$
- Kasus rekursif nya jika  $n$  lebih besar dari 1 maka fungsi memanggil dirinya sendiri dengan nilai  $n - 1$  dan mengambilkan hasil dengan  $n$  dan di return
- Maka proses ini harus terus berlanjut hingga mencapai ke basecase

Proses program tersebut berjalan dapat dilihat dari gambar dibawah ini:

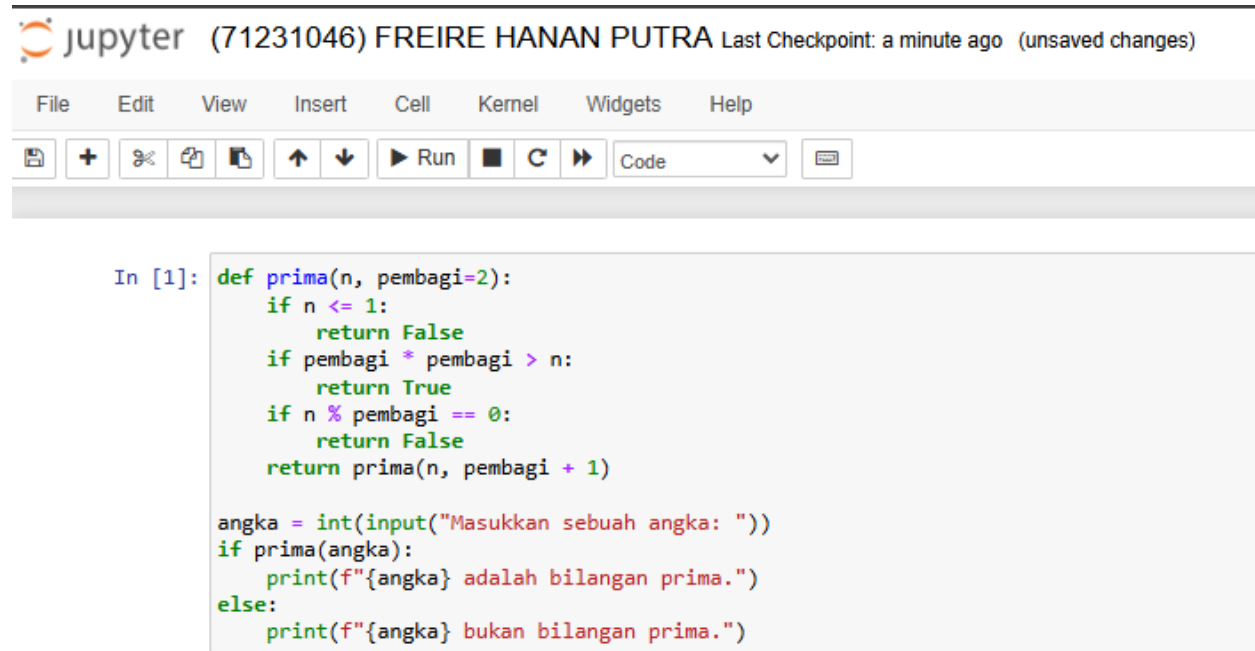
```
1.
2. calc_factorial(4)           # 1st call with 4
3. 4 * calc_factorial(3)       # 2nd call with 3
4. 4 * 3 * calc_factorial(2)   # 3rd call with 2
5. 4 * 3 * 2 * calc_factorial(1) # 4th call with 1
6. 4 * 3 * 2 * 1               # return from 4th call as number=1
7. 4 * 3 * 2                   # return from 3rd call
8. 4 * 6                       # return from 2nd call
9. 24                          # return from 1st call
```

## BAGIAN 2: LATIHAN MANDIRI (60%)

Link Github: [https://github.com/Freirehnn23/prak\\_alpro\\_week14](https://github.com/Freirehnn23/prak_alpro_week14)

### SOAL 1

Code:



The screenshot shows a Jupyter Notebook interface with the title '(71231046) FREIRE HANAN PUTRA'. The last checkpoint is noted as 'a minute ago (unsaved changes)'. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar contains icons for saving, adding cells, undo, redo, running, and other standard Jupyter actions. The code cell contains the following Python code:

```
In [1]: def prima(n, pembagi=2):
        if n <= 1:
            return False
        if pembagi * pembagi > n:
            return True
        if n % pembagi == 0:
            return False
        return prima(n, pembagi + 1)

        angka = int(input("Masukkan sebuah angka: "))
        if prima(angka):
            print(f"{angka} adalah bilangan prima.")
        else:
            print(f"{angka} bukan bilangan prima.")
```

Penerapan:

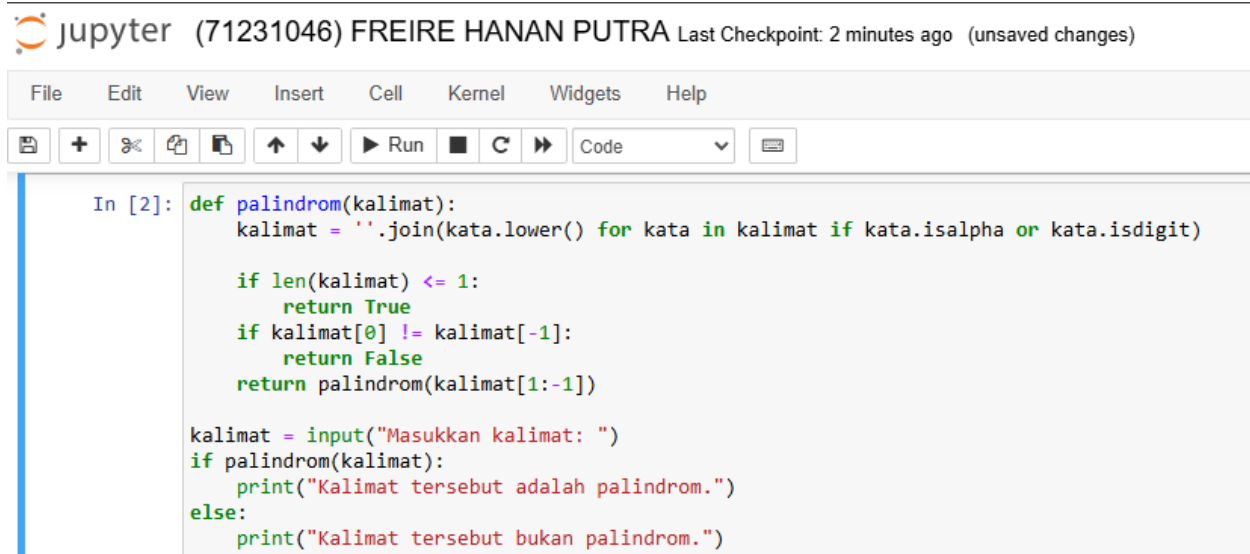
```
Masukkan sebuah angka: 3
3 adalah bilangan prima.
```

Penjelasan:

- Pertama buat fungsi dengan parameter `n` dan `pembagi` yang default nya adalah 2
- Setelah itu buat percabangan jika `n` kurang dari sama dengan 1 return false. Karena bilangan prima harus lebih dari 1 bilangan bulat positif
- Setelah itu buat basis dengan cara jika `pembagi` dikali dengan `pembagi` lebih dari `n` maka return true
- Setelah itu periksa `pembagi` dengan cara jika `n` modulus `pembagi` sama dengan 0 maka return false
- Setelah itu buat worstcase rekursif nya dengan cara return fungsi `prima n, pembagi+1` untuk memeriksa `pembagi` selanjutnya
- Setelah itu buat inputan dan di cek jika true maka print angka adalah bilangan prima
- Jika tidak maka print angka bukan bilangan prima

## SOAL 2

Code:



The screenshot shows a Jupyter Notebook window titled "jupyter (71231046) FREIRE HANAN PUTRA" with a status bar indicating "Last Checkpoint: 2 minutes ago (unsaved changes)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The code cell, labeled "In [2]:", contains a recursive Python function named "palindrom" that checks if a string is a palindrome. The function converts the string to lowercase, removes non-alphanumeric characters, and compares the first and last characters. If they match, it recursively checks the substring from the second to the second-to-last character. If they don't match, it returns False. The function is then used to prompt the user for input and print the result.

```
In [2]: def palindrom(kalimat):
        kalimat = ''.join(kata.lower() for kata in kalimat if kata.isalpha or kata.isdigit)

        if len(kalimat) <= 1:
            return True
        if kalimat[0] != kalimat[-1]:
            return False
        return palindrom(kalimat[1:-1])

        kalimat = input("Masukkan kalimat: ")
        if palindrom(kalimat):
            print("Kalimat tersebut adalah palindrom.")
        else:
            print("Kalimat tersebut bukan palindrom.")
```

Penerapan:

```
Masukkan kalimat: kasur rusak
Kalimat tersebut adalah palindrom.
```

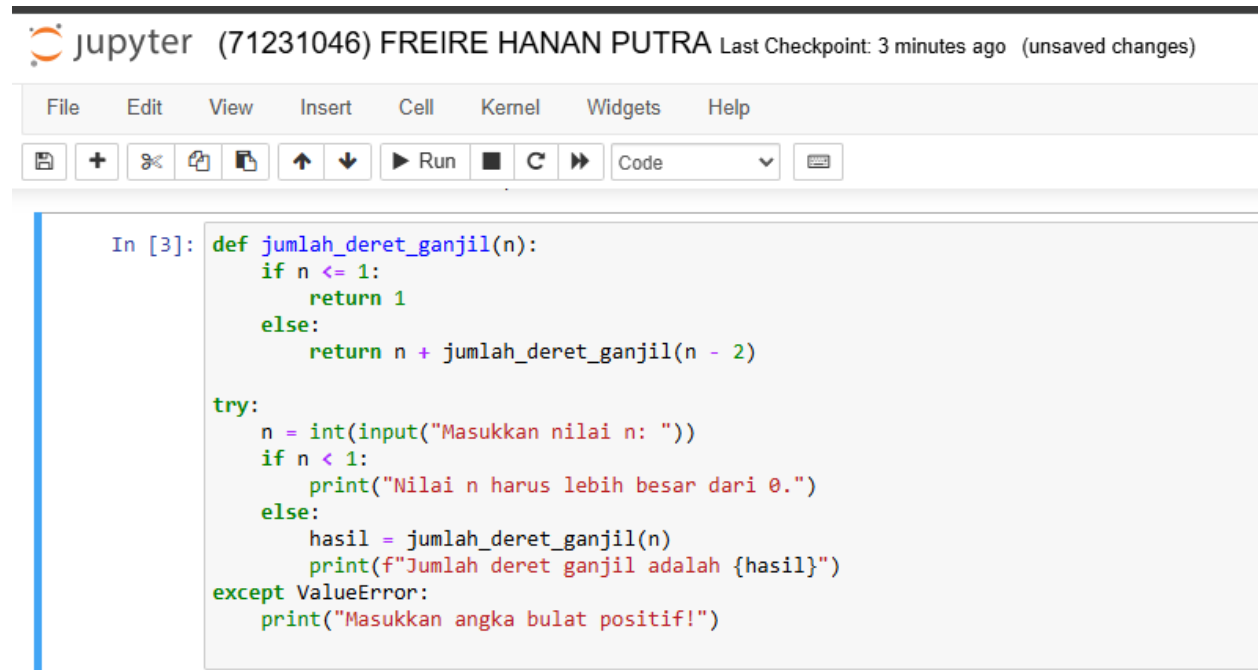
Penjelasan:

- Program memeriksa apakah input dari pengguna merupakan kalimat palindrom (dibaca sama dari depan dan belakang).
- Setelah itu semua huruf diubah menjadi huruf kecil dan hanya karakter huruf serta angka yang diperhitungkan, karakter lain diabaikan.
- Pemeriksaan ini dilakukan secara rekursif yaitu dengan cara huruf pertama dibandingkan dengan huruf terakhir.
- Jika semua pasangan karakter dari luar ke dalam cocok, maka kalimat dianggap palindrom.
- Program memberikan output yang menyatakan apakah kalimat tersebut palindrom atau bukan.



## SOAL 3

Code:



The screenshot shows a Jupyter Notebook window titled "jupyter (71231046) FREIRE HANAN PUTRA" with a last checkpoint of 3 minutes ago. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for saving, adding cells, undo, redo, and running code. The code cell contains the following Python code:

```
In [3]: def jumlah_deret_ganjil(n):
        if n <= 1:
            return 1
        else:
            return n + jumlah_deret_ganjil(n - 2)

    try:
        n = int(input("Masukkan nilai n: "))
        if n < 1:
            print("Nilai n harus lebih besar dari 0.")
        else:
            hasil = jumlah_deret_ganjil(n)
            print(f"Jumlah deret ganjil adalah {hasil}")
    except ValueError:
        print("Masukkan angka bulat positif!")
```

Penerapan:


```
Masukkan nilai n: 5
Jumlah deret ganjil adalah 9
```

Penjelasan:













- Program bagian pertama menghitung jumlah deret bilangan ganjil secara rekursif mulai dari 1 hingga nilai  $n$  (jika  $n$  adalah ganjil).
- Jika  $n$  kurang dari atau sama dengan 1, fungsi berhenti dan mengembalikan 1 sebagai nilai dasar.
- Jika  $n$  lebih dari 1, fungsi akan menjumlahkan  $n$  dengan hasil pemanggilan dirinya sendiri dengan nilai  $n - 2$ , sehingga hanya bilangan ganjil yang dijumlahkan.
- Setelah itu buat inputan dari pengguna dicek supaya bilangan tersebut merupakan bilangan bulat positif.
- Step terakhir program mencetak hasil jumlah deret ganjil jika input valid, atau memberikan pesan kesalahan jika input tidak valid.

## SOAL 4

Code:

 jupyter (71231046) FREIRE HANAN PUTRA Last Checkpoint: 4 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

        Run    Code 

```
In [4]: def jumlah_digit_rekursif(n):
        if n < 10:
            return n
        else:
            return n % 10 + jumlah_digit_rekursif(n // 10)

        try:
            angka = int(input("Masukkan bilangan: "))
            hasil = jumlah_digit_rekursif(angka)
            print(f"Jumlah digit dari {angka} adalah {hasil}")
        except ValueError:
            print("Masukkan bilangan bulat yang valid.")
```

Penerapan:

```
Masukkan bilangan: 234
Jumlah digit dari 234 adalah 9
```

Penjelasan:

- Program menghitung jumlah dari seluruh digit sebuah bilangan bulat menggunakan fungsi rekursif.
- Jika angka hanya satu digit (kurang dari 10), nilai tersebut langsung dikembalikan.
- Jika lebih dari satu digit, fungsi mengambil digit terakhir (sisa bagi 10), lalu menjumlahkannya dengan hasil pemanggilan fungsi yang sama terhadap sisa digit (pembagian bilangan dengan 10).
- Input pengguna harus berupa bilangan bulat valid. Jika tidak maka print pesan error
- Program mencetak total jumlah digit dari bilangan yang dimasukkan

## SOAL 5

Code:

 jupyter (71231046) FREIRE HANAN PUTRA Last Checkpoint: 5 minutes ago (unsaved changes)

```
File Edit View Insert Cell Kernel Widgets Help
[Icons] [Run] [Code]
In [5]: def kombinasi(n, k):
        if k == 0 or k == n:
            return 1
        else:
            return kombinasi(n - 1, k - 1) + kombinasi(n - 1, k)

        n = int(input("Masukkan nilai n: "))
        k = int(input("Masukkan nilai k: "))
        hasil = kombinasi(n, k)
        print(f"Hasil Kombinasinya adalah {hasil}")
```

Penerapan:

```
Masukkan nilai n: 4
Masukkan nilai k: 2
Hasil Kombinasinya adalah 6
```

Penjelasan:

- Program menghitung nilai kombinasi dari  $n$  ambil  $k$  (ditulis sebagai  $C(n, k)$ ) menggunakan rumus rekursif.
- Jika  $k$  sama dengan 0 atau  $k$  sama dengan  $n$ , hasil kombinasi adalah 1 (kasus dasar).
- Jika tidak, fungsi menjumlahkan dua pemanggilan rekursif: kombinasi dari  $(n-1, k-1)$  dan  $(n-1, k)$ , sesuai dengan rumus kombinasi Pascal.
- Program menerima input nilai  $n$  dan  $k$  dari pengguna, lalu menghitung dan menampilkan hasilnya.