



# Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	<71231051>
Nama Lengkap	<Amida A Ronsumbre>
Minggu ke / Materi	13/ Fungsi Rekursif

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.

PROGRAM STUDI INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI  
UNIVERSITAS KRISTEN DUTA WACANA  
YOGYAKARTA  
2024

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

Pada bagian ini, tuliskan kembali semua materi yang telah anda pelajari minggu ini. Sesuaikan penjelasan anda dengan urutan materi yang telah diberikan di saat praktikum. Penjelasan anda harus dilengkapi dengan contoh, gambar/ilustrasi, contoh program (source code) dan outputnya. Idealnya sekitar 5-6 halaman.

### MATERI 1

#### ➤ 13.3.1 Pengertian Rekursif

Fungsi rekursif adalah fungsi yang berisi dirinya sendiri atau fungsi yang mendefinisikan dirinya sendiri. Fungsi rekursif merupakan fungsi matematis yang berulang dan memiliki pola yang terstruktur, namun biasanya fungsi ini perlu diperhatikan agar fungsi ini dapat berhenti dan tidak menghabiskan memori. Fungsi ini akan terus berjalan sampai kondisi berhenti terpenuhi, oleh karena itu dalam sebuah fungsi rekursif perlu terdapat 2 blok penting, yaitu blok yang menjadi titik berhenti dari sebuah proses rekursif dan blok yang memanggil dirinya sendiri.

#### ➤ 13.3.2 Kelebihan dan Kekurangan

##### ➤ Keunggulan Fungsi Rekursif

1. Kode Program Lebih Singkat dan Elegan\*\*: Fungsi rekursif memungkinkan penulisan kode yang lebih ringkas dan mudah dipahami.

2. Memecah Masalah Kompleks\*\*: Rekursi memudahkan pemecahan masalah besar menjadi submasalah kecil yang serupa dengan masalah asli.

##### ➤ Kelemahan Fungsi Rekursif

1. Memakan Memori Lebih Besar: Setiap pemanggilan fungsi memerlukan ruang memori tambahan, yang bisa menyebabkan penggunaan memori tinggi.

2. Kurang Efisien dan Lambat: Overhead pemanggilan fungsi berulang kali mengurangi efisiensi dan kecepatan dibandingkan pendekatan iteratif.

3. Sulit Debugging dan Dimengerti: Rekursi membuat proses debugging lebih rumit dan logikanya sering sulit dipahami, terutama jika tidak jelas atau tidak tepat.

#### ➤ 13.3.3 Bentuk Umum dan Studi Kasus

Proses Perhitungan

Panggilan Pertama: faktorial(4) memanggil faktorial(3) dan mengalikan hasilnya dengan 4.

Panggilan Kedua: faktorial(3) memanggil faktorial(2) dan mengalikan hasilnya dengan 3.

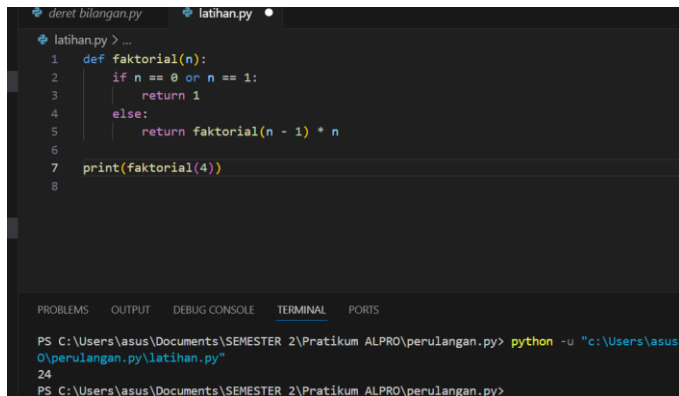
Panggilan Ketiga: faktorial(2) memanggil faktorial(1) dan mengalikan hasilnya dengan 2.

Panggilan Keempat: faktorial(1) mengembalikan 1 karena  $n == 1$ .

Penyelesaian dan Pengembalian Nilai faktorial(2) mengembalikan  $1 * 2 = 2$ .

faktorial(3) mengembalikan  $2 * 3 = 6$ . faktorial(4) mengembalikan  $6 * 4 = 24$ .

Hasil akhir yang ditampilkan adalah 24.



```
latihan.py > ...
1 def faktorial(n):
2     if n == 0 or n == 1:
3         return 1
4     else:
5         return faktorial(n - 1) * n
6
7 print(faktorial(4))
8

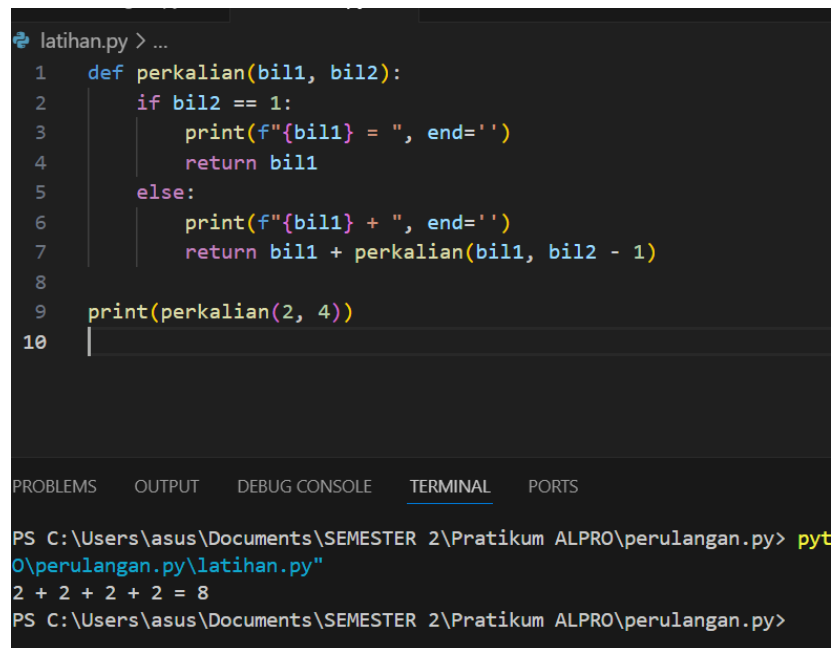
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> python -u "c:\Users\asus\
0\perulangan.py\latihan.py"
24
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py>
```

## MATERI 2

Penjelasan materi 2, dst... sesuai format ini.

- Kegiatan Pratikum
  - 13.4.1 Problem dan Solusi 1

### Kasus 13.1



```
latihan.py > ...
1 def perkalian(bil1, bil2):
2     if bil2 == 1:
3         print(f"{bil1} = ", end='')
4         return bil1
5     else:
6         print(f"{bil1} + ", end='')
7         return bil1 + perkalian(bil1, bil2 - 1)
8
9 print(perkalian(2, 4))
10

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> pyth
0\perulangan.py\latihan.py"
2 + 2 + 2 + 2 = 8
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py>
```

**Definisi Fungsi Rekursif:** Fungsi perkalian mendefinisikan basis rekursif dan langkah rekursif untuk menghentikan rekursi dan melanjutkan perhitungan.

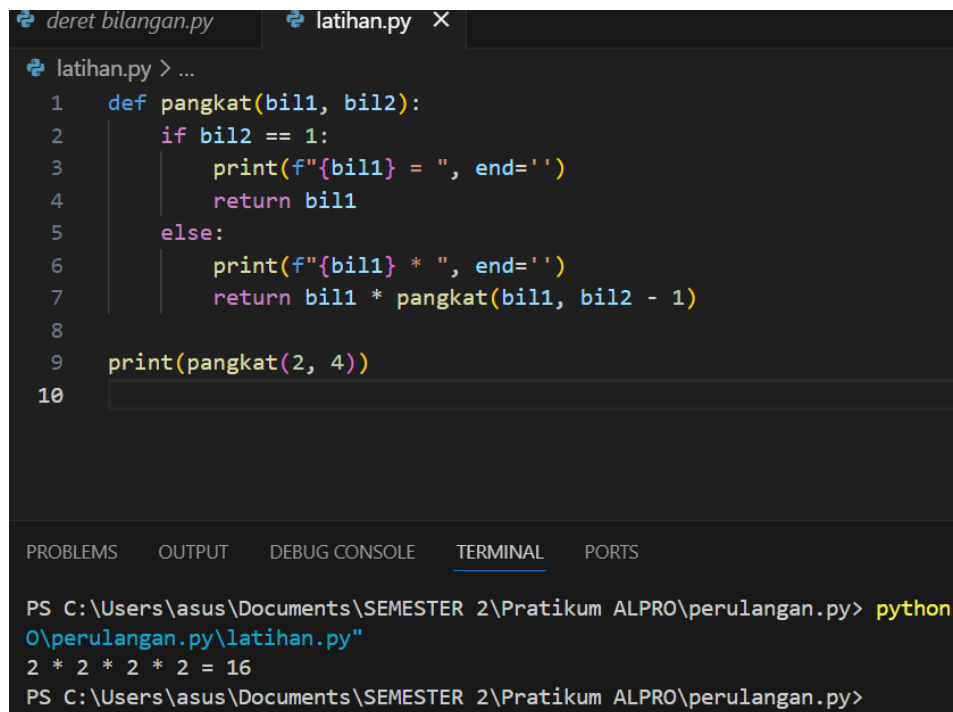
**Basis Rekursif:** Jika bilangan kedua (bil2) adalah 1, fungsi mengembalikan bilangan pertama (bil1) dan mencetak hasil akhir.

Langkah Rekursif: Jika bilangan kedua lebih besar dari 1, fungsi mencetak bilangan pertama dan memanggil dirinya sendiri dengan bilangan kedua dikurangi 1. Hasil dari pemanggilan ini ditambahkan dengan bilangan pertama.

Fungsi rekursif perkalian mendefinisikan basis dan langkah rekursif untuk mengalikan dua bilangan menggunakan penjumlahan berulang. Ketika bilangan kedua mencapai 1, fungsi berhenti dan mengembalikan hasil akhir. Proses ini mencetak langkah-langkah penjumlahan yang dilakukan dan menghasilkan hasil akhir yang benar.

- 13.4.2 Problem dan Solusi 2

### Kasus 13.2



```
deret_bilangan.py  latihan.py X
latihan.py > ...
1  def pangkat(bil1, bil2):
2      if bil2 == 1:
3          print(f"{bil1} = ", end='')
4          return bil1
5      else:
6          print(f"{bil1} * ", end='')
7          return bil1 * pangkat(bil1, bil2 - 1)
8
9  print(pangkat(2, 4))
10

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> python
O\perulangan.py\latihan.py
2 * 2 * 2 * 2 = 16
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py>
```

Definisi Fungsi: Fungsi pangkat menerima dua parameter bil1 dan bil2.

Basis Rekursif: Jika bil2 adalah 1, fungsi mengembalikan bil1 dan mencetak hasil akhir.

Langkah Rekursif: Jika bil2 lebih besar dari 1, fungsi mencetak bil1 dan memanggil dirinya sendiri dengan bil2 - 1, kemudian mengalikan hasilnya dengan bil1.

pangkat(2, 4) memanggil pangkat(2, 3) -> pangkat(2, 2) -> pangkat(2, 1).

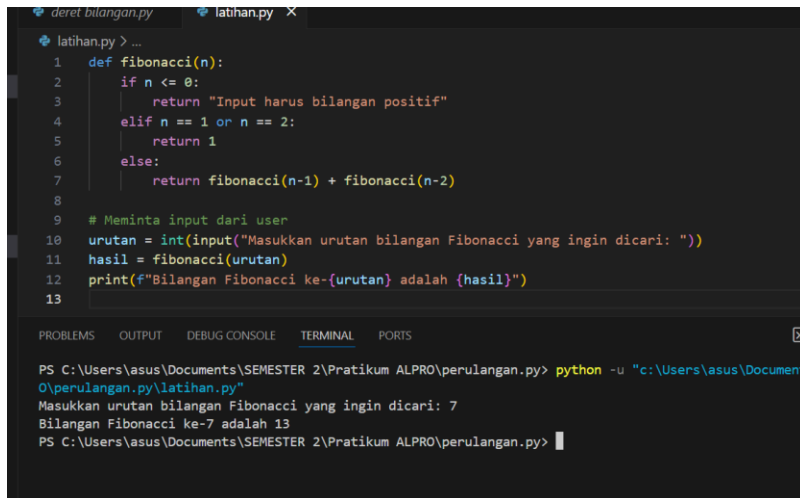
Setiap langkah mencetak bil1 \* .

Pada pangkat(2, 1), mencetak 2 = dan mengembalikan 2.

Hasil akhirnya adalah 2 \* 2 \* 2 \* 2 = 16.

- 13.4.3 Problem dan Solusi 3

### Kasus 13.3



```
1 def fibonacci(n):
2     if n <= 0:
3         return "Input harus bilangan positif"
4     elif n == 1 or n == 2:
5         return 1
6     else:
7         return fibonacci(n-1) + fibonacci(n-2)
8
9 # Meminta input dari user
10 urutan = int(input("Masukkan urutan bilangan Fibonacci yang ingin dicari: "))
11 hasil = fibonacci(urutan)
12 print(f"Bilangan Fibonacci ke-{urutan} adalah {hasil}")
13
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> python -u "c:\Users\asus\Documents\perulangan.py\latihan.py"
Masukkan urutan bilangan Fibonacci yang ingin dicari: 7
Bilangan Fibonacci ke-7 adalah 13
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py>
```

**Definisi Fungsi:** Fungsi fibonacci menerima satu parameter  $n$  yang merupakan urutan bilangan Fibonacci yang ingin dicari.

**Basis Rekursif:** Jika  $n$  adalah 1 atau 2, fungsi mengembalikan 1 karena dua bilangan pertama dalam deret Fibonacci adalah 1.

**Langkah Rekursif:** Jika  $n$  lebih besar dari 2, fungsi memanggil dirinya sendiri dengan  $n-1$  dan  $n-2$  dan menjumlahkan hasilnya.

Input: 7

Output: Bilangan Fibonacci ke-7 adalah 13

Dengan program ini, Tini dapat dengan mudah mencari bilangan pada urutan tertentu dalam deret Fibonacci tanpa harus menghitung dari awal setiap kali.

- 13.4.4 Problem dan Solusi 4

### Kasus 13.4

```
deret bilangan.py x latihan.py ●
C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py\deret bilangan.py
1 def toBasis(n, base):
2     convertString = "0123456789ABCDEF"
3     if n < base:
4         return convertString[n]
5     else:
6         return toBasis(n // base, base) + convertString[n % base]
7
8 print("Silahkan masukkan bilangan dan basis:")
9 angka = int(input("Bilangan: "))
10 basis = int(input("Basis (2/8/16): "))
11
12 if basis not in [2, 8, 16]:
13     print("Basis yang diperbolehkan hanya 2, 8, atau 16.")
14 else:
15     hasil = toBasis(angka, basis)
16     print(f"Bilangan {angka} dalam basis {basis} adalah: {hasil}")
17
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> python -u "c:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py\latihan.py"
Silahkan masukkan bilangan dan basis:
Bilangan: 8
Basis (2/8/16): 4
Basis yang diperbolehkan hanya 2, 8, atau 16.
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> 
```

**Definisi Fungsi:** Fungsi toBasis menerima dua parameter, n (bilangan dalam basis 10) dan base (basis tujuan).

**Basis Rekursif:** Jika n kurang dari base, fungsi mengembalikan karakter yang sesuai dari convertString berdasarkan nilai n.

**Langkah Rekursif:** Jika n lebih besar atau sama dengan base, fungsi memanggil dirinya sendiri dengan n // base dan menambahkan karakter yang sesuai dari convertString berdasarkan nilai n % base.

**Input dan Validasi:** Program meminta pengguna memasukkan bilangan dan basis yang diinginkan, lalu memastikan basis yang dimasukkan adalah 2, 8, atau 16.

**Output:** Program mencetak hasil konversi bilangan ke basis yang diinginkan.

**Contoh Penggunaan**

Input:

Bilangan: 9

Basis: 8

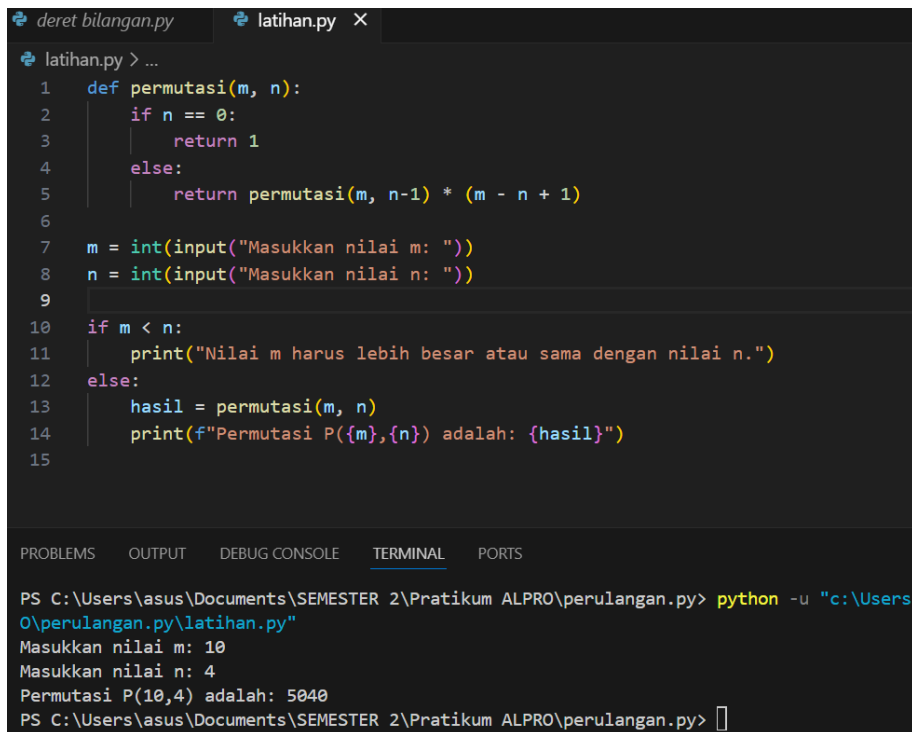
Output:

Bilangan 9 dalam basis 8 adalah: 11

Dengan program ini, pengguna dapat mengkonversi bilangan dari basis 10 ke basis 2, 8, atau 16 dengan mudah menggunakan fungsi rekursif.

- 13.4.5 Problem dan Solusi 5

### Kasus 13.5



```
deret_bilangan.py  latihan.py X
latihan.py > ...
1 def permutasi(m, n):
2     if n == 0:
3         return 1
4     else:
5         return permutasi(m, n-1) * (m - n + 1)
6
7 m = int(input("Masukkan nilai m: "))
8 n = int(input("Masukkan nilai n: "))
9
10 if m < n:
11     print("Nilai m harus lebih besar atau sama dengan nilai n.")
12 else:
13     hasil = permutasi(m, n)
14     print(f"Permutasi P({m},{n}) adalah: {hasil}")
15

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> python -u "c:\Users\
O\perulangan.py\latihan.py"
Masukkan nilai m: 10
Masukkan nilai n: 4
Permutasi P(10,4) adalah: 5040
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> 
```

Definisi Fungsi: Fungsi permutasi menerima dua parameter, m (batas atas) dan n (batas bawah).

Basis Rekursif: Jika n adalah 0, fungsi mengembalikan 1 karena  $m!(m-0)! = 1(m-0)!m! = 1$ .

Langkah Rekursif: Jika n lebih besar dari 0, fungsi memanggil dirinya sendiri dengan parameter m dan n-1, kemudian mengalikan hasilnya dengan (m - n + 1).

Contoh Penggunaan:

Input:

m: 10

n: 4

Output:

Permutasi  $P(10,4)$  adalah: 5040

Dengan program ini, pengguna dapat menghitung permutasi secara rekursif untuk nilai m dan n yang diinginkan.

## BAGIAN 2: LATIHAN MANDIRI (60%)

Pada bagian ini anda menuliskan jawaban dari soal-soal Latihan Mandiri yang ada di modul praktikum. Jawaban anda harus disertai dengan source code, penjelasan dan screenshot output.

### SOAL 1

**Definisi Fungsi:** Fungsi `is_prime` menerima dua parameter: `n` (bilangan yang akan diperiksa) dan `divisor` (pembagi yang akan digunakan dalam pemeriksaan rekursif).

**Inisialisasi Pembagi:** Jika `divisor` adalah `None`, maka diinisialisasi menjadi `n - 1` pada panggilan pertama.

**Kasus Basis:**

Jika `n` kurang dari atau sama dengan 1, bilangan tersebut bukan bilangan prima.

Jika `divisor` adalah 1, maka `n` adalah bilangan prima.

**Kasus Rekursif:**

Jika `n` habis dibagi oleh `divisor`, maka `n` bukan bilangan prima.

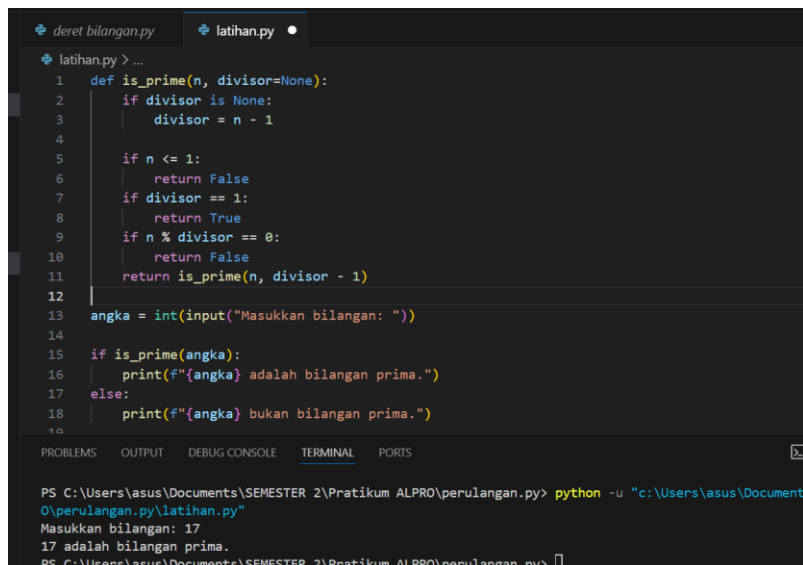
Jika tidak, fungsi memanggil dirinya sendiri dengan `divisor - 1`.

**Contoh Penggunaan**

Input: 17

Output: 17 adalah bilangan prima.

Program ini akan membantu Vidi untuk memeriksa apakah suatu bilangan adalah bilangan prima dengan mudah menggunakan fungsi rekursif.



```
deret bilangan.py • latihan.py •
latihan.py > ...
1 def is_prime(n, divisor=None):
2     if divisor is None:
3         divisor = n - 1
4
5     if n <= 1:
6         return False
7     if divisor == 1:
8         return True
9     if n % divisor == 0:
10        return False
11    return is_prime(n, divisor - 1)
12
13 angka = int(input("Masukkan bilangan: "))
14
15 if is_prime(angka):
16     print(f"{angka} adalah bilangan prima.")
17 else:
18     print(f"{angka} bukan bilangan prima.")
19
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> python -u "c:\Users\asus\Documents\perulangan.py\latihan.py"
Masukkan bilangan: 17
17 adalah bilangan prima.
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> |
```



## Soal 2

Fungsi `is_palindrome` menerima sebuah string kalimat.

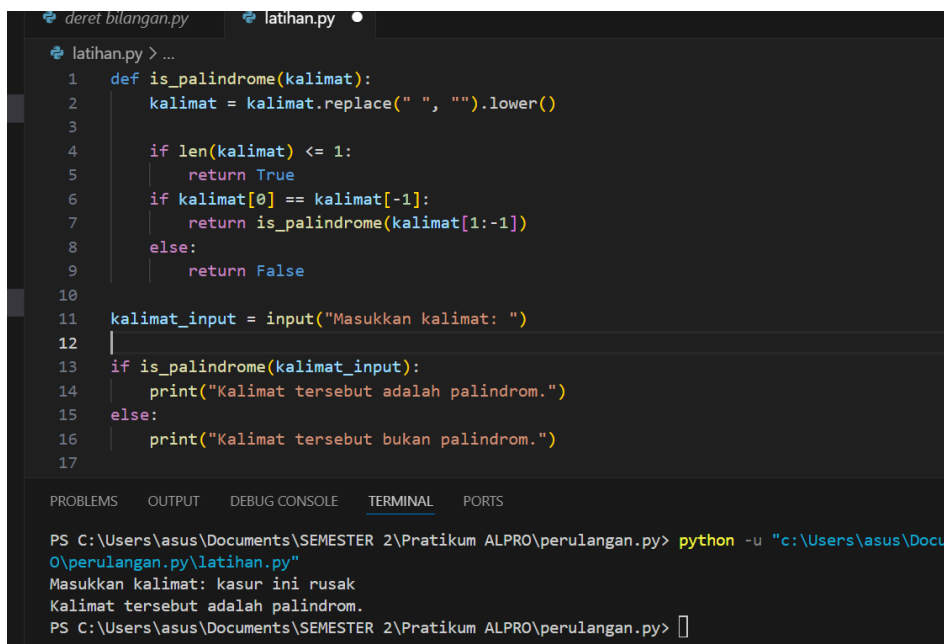
Langkah pertama adalah menghilangkan spasi dan mengubah semua huruf menjadi huruf kecil menggunakan metode `replace()` dan `lower()`.

Basis rekursif: Jika panjang kalimat adalah 0 atau 1, maka kalimat tersebut adalah palindrom.

Pengecekan karakter pertama dan terakhir: Jika karakter pertama sama dengan karakter terakhir, fungsi akan memanggil dirinya sendiri untuk memeriksa bagian dalam kalimat (dengan menghilangkan karakter pertama dan terakhir).

Jika kedua karakter tidak sama, maka kalimat bukan palindrom.

Program mencetak hasil berdasarkan hasil pengecekan.



```
def is_palindrome(kalimat):
    kalimat = kalimat.replace(" ", "").lower()

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

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

PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> python -u "c:\Users\asus\Documents\perulangan.py\latihan.py"

Masukkan kalimat: kasur ini rusak

Kalimat tersebut adalah palindrom.

PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py>

## Soal 3

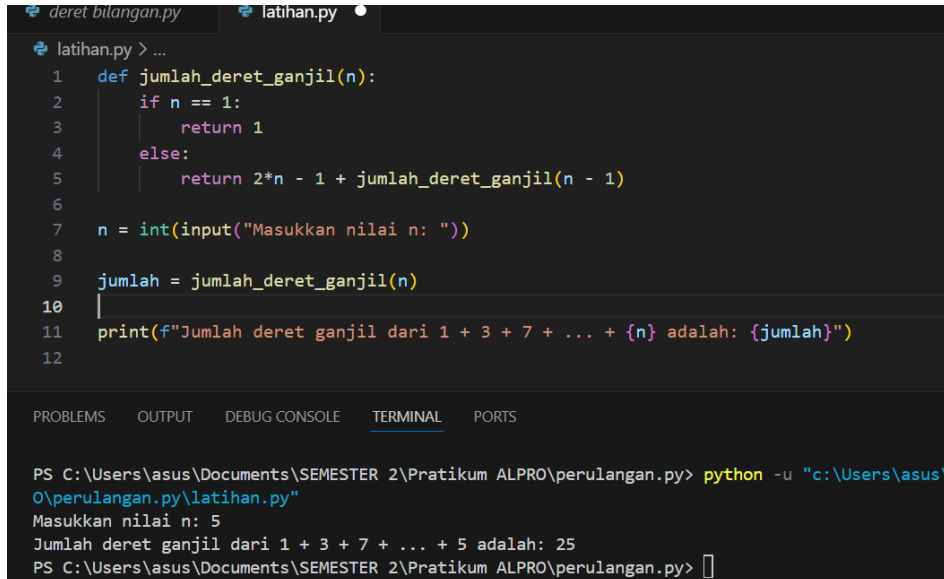
### Penjelasan

Fungsi `jumlah_deret_ganjil` menerima sebuah bilangan bulat  $n$  sebagai parameter.

Pada panggilan rekursif, fungsi mengurangi  $n$  dengan 1 dan menambahkan 2 kali nilai  $n$  dikurangi 1 (karena setiap bilangan ganjil adalah  $2n - 1$ ).

Ketika nilai  $n$  adalah 1, fungsi mengembalikan 1, sebagai basis rekursif, karena 1 adalah bilangan ganjil terkecil.

Program meminta pengguna untuk memasukkan nilai  $n$ , menghitung jumlah deret ganjil menggunakan fungsi rekursif, dan kemudian menampilkan hasilnya.



```
deret_bilangan.py  latihan.py
latihan.py > ...
1  def jumlah_deret_ganjil(n):
2      if n == 1:
3          return 1
4      else:
5          return 2*n - 1 + jumlah_deret_ganjil(n - 1)
6
7  n = int(input("Masukkan nilai n: "))
8
9  jumlah = jumlah_deret_ganjil(n)
10
11 print(f"Jumlah deret ganjil dari 1 + 3 + 7 + ... + {n} adalah: {jumlah}")
12

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> python -u "c:\Users\asus\O\perulangan.py\latihan.py"
Masukkan nilai n: 5
Jumlah deret ganjil dari 1 + 3 + 7 + ... + 5 adalah: 25
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> 
```

## SOAL 4

Fungsi `jumlah_digit` menerima sebuah bilangan bulat  $n$  sebagai parameter.

Pada panggilan rekursif, fungsi memanggil dirinya sendiri dengan parameter  $n // 10$  untuk mendapatkan bilangan  $n$  tanpa digit terakhir, sambil menambahkan digit terakhir ( $n \% 10$ ) ke jumlah total.

Ketika nilai  $n$  adalah 0, fungsi mengembalikan 0, sebagai basis rekursif, karena tidak ada digit yang tersisa untuk dijumlahkan.

Program meminta pengguna untuk memasukkan sebuah bilangan, menghitung jumlah digit dari bilangan tersebut menggunakan fungsi rekursif, dan kemudian menampilkan hasilnya.

Dengan program ini, Anda dapat dengan mudah menghitung jumlah digit dari suatu bilangan menggunakan fungsi rekursif.

```
latihan.py > ...
1 def jumlah_digit(n):
2     if n == 0:
3         return 0
4     else:
5         return n % 10 + jumlah_digit(n // 10)
6
7 bilangan = int(input("Masukkan bilangan: "))
8
9 jumlah = jumlah_digit(bilangan)
10
11 print(f"Jumlah digit dari {bilangan} adalah: {jumlah}")
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> python -u "c:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py\latihan.py"
Masukkan bilangan: 234
Jumlah digit dari 234 adalah: 9
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> 
```

## SOAL 5

Untuk menghitung kombinasi, kita dapat menggunakan rumus matematika

$C(n,k) = \frac{n!}{k!(n-k)!}$   $C(n,k) = \frac{n!}{k!(n-k)!}$ , di mana  $n$  adalah jumlah elemen total,  $k$  adalah jumlah elemen

yang dipilih, dan  $n!$  merupakan faktorial dari  $n$ . Untuk menghitung kombinasi, kita dapat

menggunakan rumus matematika  $C(n,k) = \frac{n!}{k!(n-k)!}$   $C(n,k) = \frac{n!}{k!(n-k)!}$ , di mana  $n$  adalah jumlah elemen total,  $k$  adalah jumlah elemen yang dipilih, dan  $n!$  merupakan faktorial dari  $n$ .

Dalam program ini, kita memiliki dua fungsi:

faktorial(n): Fungsi ini digunakan untuk menghitung faktorial dari suatu bilangan  $n$  menggunakan pendekatan rekursif.

kombinasi(n, k): Fungsi ini menghitung kombinasi  $C(n,k)$  menggunakan rumus kombinasi matematika. Dengan program ini, kita dapat dengan mudah menghitung kombinasi  $C(n,k)$  menggunakan fungsi rekursif.

```
deret bilangan.py latihan.py
latihan.py > ...
1 def faktorial(n):
2     if n == 0 or n == 1:
3         return 1
4     else:
5         return n * faktorial(n - 1)
6
7 def kombinasi(n, k):
8     if k > n:
9         return "Kombinasi tidak bisa dihitung karena k lebih besar dari n."
10    else:
11        return faktorial(n) // (faktorial(k) * faktorial(n - k))
12
13 n = int(input("Masukkan nilai n: "))
14 k = int(input("Masukkan nilai k: "))
15
16 hasil_kombinasi = kombinasi(n, k)
17
18 print(f"Kombinasi C({n}, {k}) adalah: {hasil_kombinasi}")
19

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> python -u "c:\Users\asus\Documents\perulangan.py\latihan.py"
Masukkan nilai n: 5
Masukkan nilai k: 2
Kombinasi C(5, 2) adalah: 10
PS C:\Users\asus\Documents\SEMESTER 2\Pratikum ALPRO\perulangan.py> 
```