



# Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

|                           |                                 |
|---------------------------|---------------------------------|
| <b>NIM</b>                | <b>71231052</b>                 |
| <b>Nama Lengkap</b>       | <b>GABRIEL SACHIO ATMADJAJA</b> |
| <b>Minggu ke / Materi</b> | <b>14 / Fungsi Rekursif</b>     |

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%)

Sumber: Modul Pratikum Algoritma dan Pemrograman, <https://shorturl.at/eezDw>

### MATERI 1

Fungsi rekursif itu fungsi yang mengulangi atau memanggil dirinya sendiri. Fungsi rekursif terdiri dari 2 struktur yaitu *best case* dan *recursive case*. *Best case* berarti bagian yang membuat fungsi itu berhenti mengulang atau memanggil dirinya, sedangkan *recursive case* itu bagian yang membuat fungsi itu memanggil dirinya sendiri. Perhatikan gambar berikut:

```
18 def rekursif(x):
19     if x == 1:                                # best case
20         print(f"{x} =", end=" ")
21         return x
22     else:                                      # recursive case
23         print(f"{x} +", end=" ")
24         return x + rekursif(x-1)
25
26 print(rekursif(3))
```

Gambar 1.1: Contoh fungsi rekursif

Dari gambar 1.1, kondisi if dan kode di dalamnya merupakan contoh dari *best case* karena hanya akan dijalankan 1 kali lalu program selesai. Lalu pada kondisi else dan kode di dalamnya merupakan contoh dari *recursive case* karena terdapat kode yang memanggil dirinya sendiri (fungsi rekursif) yaitu kode `return x + rekursif(x - 1)`. Ketika program dijalankan, hasilnya akan mirip dengan loop. Berikut outputnya:

```
PS D:\Sachio\Kampus\PrakAlPro\Pert14> & C:/Users/Acer/AppData/Local/Programs/Python/Python312/
3 + 2 + 1 = 6
```

Gambar 1.2: Output

### Perbedaan Fungsi rekursif dan Loop

- Loop itu bukan sebuah fungsi, jadi dia tidak bisa memanggil dirinya sendiri, sedangkan fungsi rekursif dapat memanggil dirinya sendiri

### Kelebihan dan Kekurangan fungsi rekursif

- Kelebihannya → kode program lebih singkat, serta masalah kompleks dapat dipecah menjadi beberapa masalah kecil.
- Kekurangannya → memakan memori yang lebih besar, mengorbankan efisiensi dan kecepatan, fungsi rekursif sulit di debugging dan kadang sulit dimengerti

**Contoh lain penggunaan fungsi rekursif:**

➤ Program Faktorial

```
28 def faktorial(n):
29     if n == 0 or n == 1:
30         return 1
31     else:
32         return faktorial(n-1) * n
33
34 print(faktorial(4))
```

Gambar 1.3: Program faktorial dengan fungsi rekursif

```
PS D:\Sachio\Kampus\PrakAlPro\Pert14> & C:/Users/Acer/AppData/Local/
24
```

Gambar 1.4: Output

## BAGIAN 2: LATIHAN MANDIRI (60%)

Source: <https://github.com/gabrielsachioa/PrakAIPro13.git>

### SOAL 1

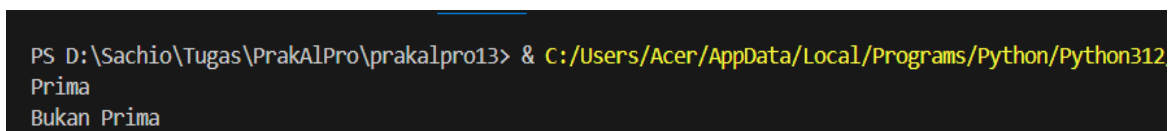


```
lat13-1.py > ...
1 def cek_bilangan_prima(n, i = 2):
2     if n <= 1:
3         return "Bukan Prima"
4     else:
5         if i > n:
6             return "Prima"
7         else:
8             if n == 2:
9                 return "Prima"
10            elif n % 2 == 1:
11                return cek_bilangan_prima(n, i + 1)
12            else:
13                return "Bukan Prima"
14
15 print(cek_bilangan_prima(19))
16 print(cek_bilangan_prima(6))
```

Gambar 2.1: Program cek bilangan prima dengan fungsi rekursif

Keterangan → intinya best casenya adalah ketika  $n \leq 1$ ,  $i > n$ , atau saat  $n == 2$ . Bagian recursive casenya itu saat  $n \% 2 == 1$ , maksudnya ketika sisa bagi  $n$  adalah 1, maka akan memanggil fungsi itu kembali dengan cara `cek_bilangan_prima(n, i + 1)`. Maksudnya itu bilangan di huruf  $n$  akan dicek oleh  $i$  (defaultnya adalah 2), apakah sampai mencapai salah satu best case... Kalau ternyata bilangan di  $n$  bisa dibagi selain bilangan itu sendiri dan 1, maka bukan prima. Jika hanya bisa dibagi bilangan itu sendiri dan 1, maka itu prima.

Output:



```
PS D:\Sachio\Tugas\PrakAIPro\prakalpro13> & C:/Users/Acer/AppData/Local/Programs/Python/Python312
Prima
Bukan Prima
```

Gambar 2.2: Output

## SOAL 2

```
lat13-2.py > ...
1 def cek_palindrom(string):
2     if len(string) == 1:
3         return "Palindrom"
4     else:
5         # len(string) > 1
6         if string == string[::-1]:
7             return cek_palindrom(string[1:-1])
8         else:
9             return "Bukan Palindrom"
10
11 print(cek_palindrom("kasurininirusak"))
12 print(cek_palindrom("babu"))
```

Gambar 2.3: Program cek palindrom dengan fungsi rekursif

Keterangan → kalau panjang string yang diinput hanya 1, maka tentu saja itu palindrom. Lalu dalam kondisi else, jika `string == string[::-1]`, maka `return cek_palindrom(string[1:-1])`. Maksudnya kode itu akan memanggil fungsi tersebut lagi dengan mengecek dari string index ke 1 dengan string index ke (-1). Dan akan berulang terus hingga mencapai best case.

Output:

```
PS D:\Sachio\Tugas\PrakAlPro\prakalpro13> & C:/Users/Acer/AppData/Local/Programs/Python/Python312/
Palindrom
Bukan Palindrom
```

Gambar 2.4: Output

## SOAL 3

```
lat13-3.py > ...
1 def jumlah_deret_ganjil(n):
2     if n < 1:
3         return 0
4     elif n % 2 != 0:
5         return n + jumlah_deret_ganjil(n - 2)
6     else:
7         return jumlah_deret_ganjil(n - 1)
8
9 print(jumlah_deret_ganjil(7))
```

Gambar 2.5: Program jumlah deret ganjil dengan fungsi rekursif

Keterangan → jika  $n < 1$ , maka return 0. Lalu kondisi elif, Jika  $n$  dibagi 2, sisanya tidak sama dengan 0, maka akan return  $n + \text{jumlah\_deret\_ganjil}(n - 2)$ , maksudnya itu setiap  $n$  bilangan ganjil, selisihnya akan dikurangi 2 untuk menghitung jumlah deret ganjilnya. Lalu jika  $n$  bukan bilangan ganjil,  $n$  akan dikurangi 1 terlebih dahulu, supaya bisa mendapat bilangan ganjilnya. Baru menghitung kode yang  $n - 2$  itu.

Output:

```
PS D:\Sachio\Tugas\PrakAlPro\prakalpro13> & C:/Users/Acer/AppData/Local/Programs/Python/Python312/python.exe
```

Gambar 2.6: Output

#### SOAL 4

```
lat13-4.py > ...
1 def jumlah_digit(n):
2     if n < 10 and not n < 0:
3         # kalau 1 digit, return digit itu
4         return n
5     else:
6         # dua atau lebih baru jumlahkan
7         return n % 10 + jumlah_digit(n // 10)
8
9 # PROGRAM UTAMA
10 print(jumlah_digit(2))
11 print(jumlah_digit(23))
12 print(jumlah_digit(234))
```

Gambar 2.7: Program jumlah digit dengan fungsi rekursif

Keterangan → dalam kondisi if, maksud kodenya itu jika  $n < 10$  / digitnya cuma 1 dan bukan bilangan negatif, maka akan return bilangan ( $n$ ) itu sendiri. Lalu dalam kondisi else, maksud  $n \% 10$  itu untuk mengecek digit lainnya (jika digit lebih dari 1) akan ditambah dengan  $\text{jumlah\_digit}(n // 10)$ . Jadi fungsi itu akan memanggil sisanya. Contohnya pada test case jika  $n$ -nya 23. Maka  $23 \% 10$  itu akan menghasilkan sisa 3, lalu akan memanggil fungsi itu lagi dengan parameternya  $n$  dibagi 10 dan dibulatkan ke bawah, sehingga nanti parameternya akan menjadi 2. Karena parameter fungsi jadi 2 dan itu masuk kondisi if, maka itu akan dihitung  $3 + 2$  dan menghasilkan output 5.

Output:

```
PS D:\Sachio\Tugas\PrakAlPro\prakalpro13> & C:/Users/Acer/AppData/Local/Programs/Python/Python312/python.exe
2
5
9
```

Gambar 2.8: Output

## SOAL 5

```
lat13-5.py > ...
1 def combination(n, r):
2     if r == 0:
3         return 1
4     elif n < r:
5         return "please enter n ≥ r ≥ 0"
6     else:
7         return n * combination(n - 1, r - 1) / r
8
9 print(combination(5, 0))
10 print(combination(1, 5))
11 print(combination(5, 2))
```

Gambar 2.9: Program hitung kombinasi dengan fungsi rekursif

Keterangan → fungsi memiliki parameter  $n$  dan  $r$ . Bagian best case pertama terjadi ketika  $r == 0$ , otomatis hasilnya 1, jadi tinggal return 1. Best case kedua terjadi kalau  $n < r$ , maka itu akan error. Lalu bagian recursive casenya itu di dalam kondisi else (jika  $n \geq r \geq 0$ ), maka akan return  $n * \text{combination}(n - 1, r - 1) / r$ . Misal test case ke 3, akan return  $5 * \text{combination}(4, 1) / 1$ . Combination(4, 1) itu adalah  $4 * \text{combination}(3, 0)$ . Karena combination(3,0) itu sendiri adalah 1, jadi program nanti akan menghitung  $5 * 4 / 2$ . Jadi 20 dibagi 2 adalah 10

Output:

```
PS D:\Sachio\Tugas\PrakAlPro\prakalpro13> & C:/Users/Acer/AppData/Local/Programs/Python/Python312/py
1
please enter n ≥ r ≥ 0
10.0
```

Gambar 2.10: Output