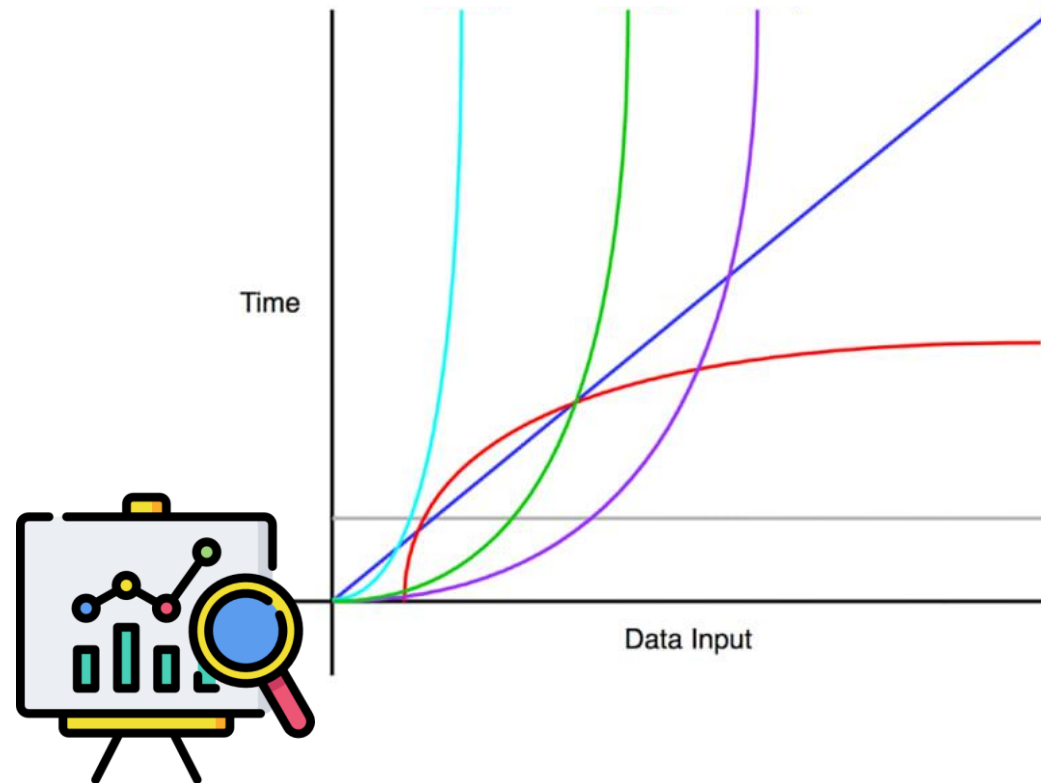


# Motivasi



# Apersepsi

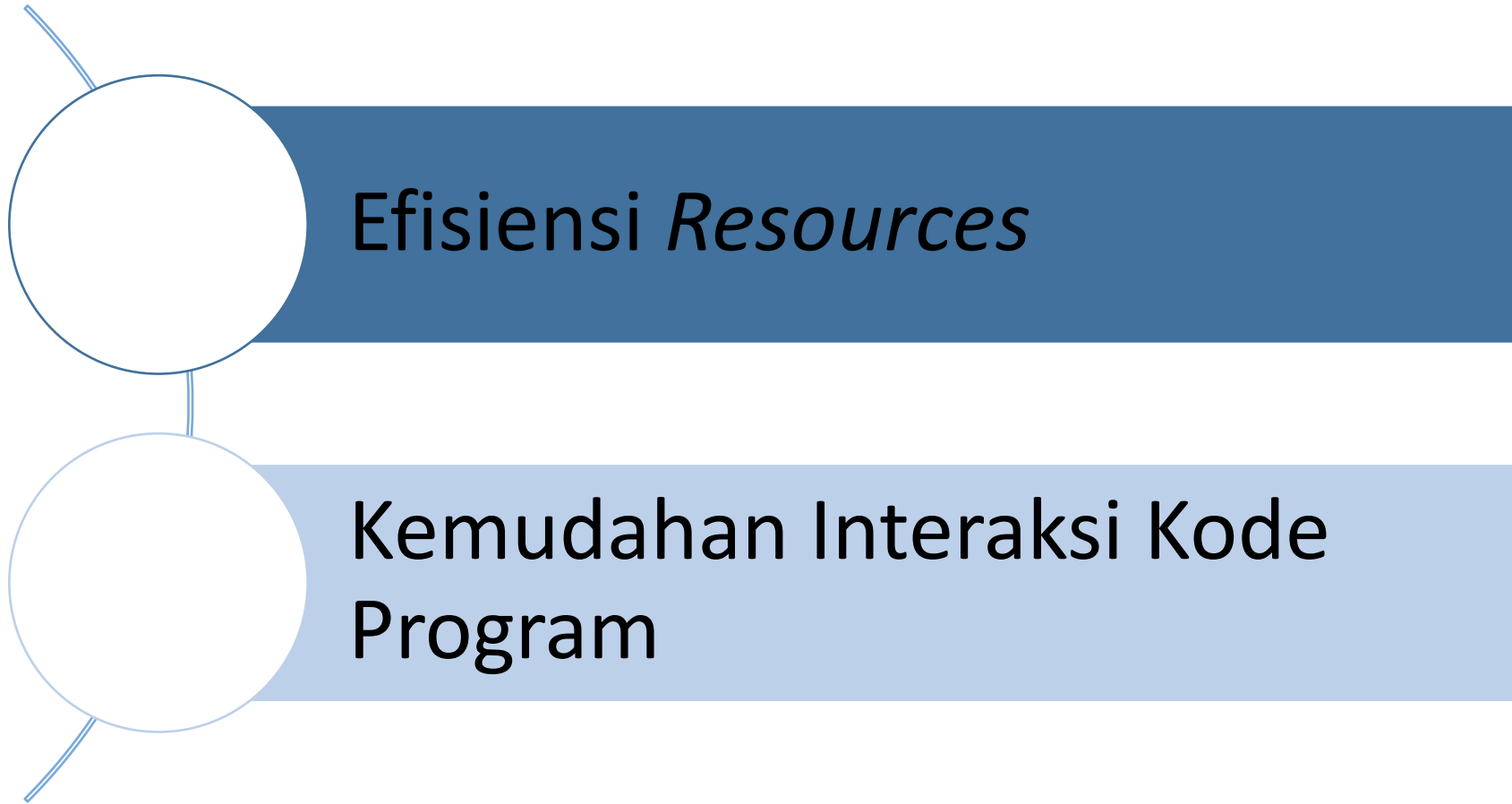


# Menggunakan Ukuran Performasi dalam Menuliskan Kode Sumber

# Ringkasan Mata Pelatihan

- Unit Kompetensi Acuan: Menulis kode dengan prinsip sesuai *guidelines* dan *best practices*
- Kode Unit Kompetensi Acuan: J.620100.016.01
- Deskripsi singkat: Mata pelatihan ini menentukan kompetensi, pengetahuan dan sikap kerja yang diperlukan dalam menerapkan penulisan kode.
- Tujuan Pembelajaran: Peserta dapat menerapkan penulisan kode yang baik agar kode tersebut dapat dirawat (*maintainability*).

# Agenda



# Efisiensi *Resources*

# Efisiensi *Resource*

Faktor-faktor yang harus diperhatikan untuk efisiensi:

1. Kemudahan kode program untuk dibaca
2. *Resource* yang dipakai untuk eksekusi kode program

Optimasi kode program dapat dilakukan dengan cara:


1. Penamaan *variable*
2. *Reusable code*
3. Optimasi logika

# Penamaan *Variable*

*Variable* yang digunakan harus mudah dibaca orang lain. *Variable* dapat ditambahkan komentar atau membuat *variable* yang mudah ditebak atau dibaca.

```
$x = "BPPTIK";  
$y = "Cikarang, Bekasi";  
print $x. ' berlokasi di ' .$y;
```

```
$lembaga = "BPPTIK";  
$lokasi = "Cikarang, Bekasi";  
print $lembaga. ' berlokasi di ' .$lokasi;
```



Penamaan *variable* menjadi **\$lembaga** dan **\$lokasi** membuat *variable* lebih mudah ditebak dan dibaca



# Reusable Code

Kode program yang digunakan berulang kali sebaiknya disimpan dalam bentuk *function* sehingga dapat digunakan kembali untuk keperluan yang lain.

```
$lembaga = "BPPTIK";  
$lokasi = "Cikarang, Bekasi";  
print $lembaga. ' berlokasi di ' .$lokasi;  
  
$lembaga2 = "Kementerian Kominfo";  
$lokasi2 = "Medan Merdeka, Jakarta";  
print $lembaga2. ' berlokasi di ' .$lokasi2;
```

```
function location($lembaga, $lokasi){  
    print $lembaga. ' berlokasi di ' .$lokasi;  
}  
  
location('BPPTIK', 'Cikarang, Bekasi');  
location('Kementerian Kominfo', 'Medan  
    Merdeka, Jakarta');
```



*Function* **location** dapat digunakan berulang kali dengan mengganti isi *variable* yang dibutuhkan

# Optimasi Logika

Dari *function* location ingin ditambahkan logika dimana apabila \$lembaga bernilai BPPTIK, maka tulisan BPPTIK yang ditampilkan berwarna biru.

```
function location($lembaga, $lokasi){  
    if($lembaga == 'BPPTIK'){  
        print '<span style="color:blue">'.$lembaga.'</span> : '.$lokasi;  
    }  
    else{  
        print '<span style="color:black">'.$lembaga.'</span> : '.$lokasi;  
    }  
}  
  
location('BPPTIK', 'Cikarang, Bekasi');  
echo "<br>";  
location('Kementerian Kominfo', 'Medan Merdeka, Jakarta');
```

```
function location($lembaga, $lokasi){  
    $color = 'black';  
    if('BPPTIK' == $lembaga){  
        $color = 'blue';  
    }  
    print '<span style="color:'.$color.'">'.$lembaga.'</span> : '.$lokasi;  
}  
  
location('BPPTIK', 'Cikarang, Bekasi');  
echo "<br>";  
location('Kementerian Kominfo', 'Medan Merdeka, Jakarta');
```

BPPTIK : Cikarang, Bekasi

Kementerian Kominfo : Medan Merdeka, Jakarta

Logika *function* dioptimasi dengan hanya satu aksi yang dilakukan, serta **if else** yang berlaku hanya satu saja

# Tips: *Source Code* yang Efisien

Berikut adalah beberapa tips untuk membuat program yang lebih efisien:

- **Manajemen *variable***  
Gunakan *variable* untuk menyimpan nilai yang akan dipanggil, usahakan tidak ada *variable* yang tidak terpakai.
- **Menyimpan hasil perhitungan kedalam *variable***  
Simpanlah hasil perhitungan ke dalam *variable* jika terdapat operasi aritmatika yang dilakukan lebih dari sekali, ini juga berlaku untuk fungsi yang mengembalikan nilai.

# Tips: *Source Code* yang Efisien (2)

- **Meminimalisir perulangan bertingkat**

Semakin banyak perulangan bertingkat yang dilakukan, maka akan semakin meningkatkan CPU time. Jika menggunakan perulangan bertingkat, batasi perulangan yang berada paling dalam dan perulangan pertama bernilai lebih besar dibanding perulangan di dalamnya.

- **Menghindari perulangan fungsi**

Jika terdapat perulangan yang memanggil fungsi yang mengembalikan nilai, jadikanlah fungsi tersebut sebuah perulangan.

# Kemudahan Interaksi Kode Program

# Kemudahan Interaksi Kode Program

Masalah yang muncul dalam pembangunan dan pengembangan program yang dilakukan lebih dari satu orang adalah kode program ditulis sesuai dengan gaya / *style* masing-masing *programmer*. Untuk memudahkan interaksi antar programmer, dibutuhkan standar atau *coding guidelines* seperti yang sudah dibahas sebelumnya.

# Kemudahan Interaksi Kode Program (2)

Manfaat yang didapatkan *programmer* jika kode program yang dibuat mengikuti *coding guidelines*:

- Kode program mudah dibaca
- Konsistensi penamaan *variable*, fungsi dan semua isi kode program
- Penamaan fungsi merepresentasikan isi di dalamnya
- Kode program dapat dimengerti oleh *programmer* lain

# Kesimpulan



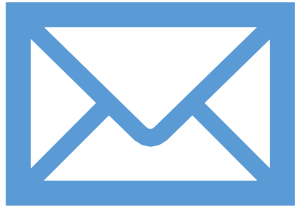
# Kesimpulan

- *Coding guidelines* dibutuhkan untuk mempermudah interaksi antar *programmer* dalam pembuatan atau *maintenance* program.
- Semakin kompleks kode program yang dibangun maka semakin besar resource yang dibutuhkan untuk menjalankan program tersebut.

# Referensi / Bacaan Lebih Lanjut

# Referensi / Bacaan Lebih Lanjut

- <https://www.kodinggen.com/tips-ngoding-membuat-source-code-yang-efisien/>



Kantor:

Balai Pelatihan dan Pengembangan  
Teknologi Informasi dan Komunikasi  
Kementerian Kominfo

Website: <https://bpptik.kominfo.go.id>

Email: [bpptik@kominfo.go.id](mailto:bpptik@kominfo.go.id)

Twitter: @bpptik

Facebook: @bpptik

Instagram: @bpptik

Google Plus: +bpptikkemkominfo

# Terima Kasih

## BPPTIK