

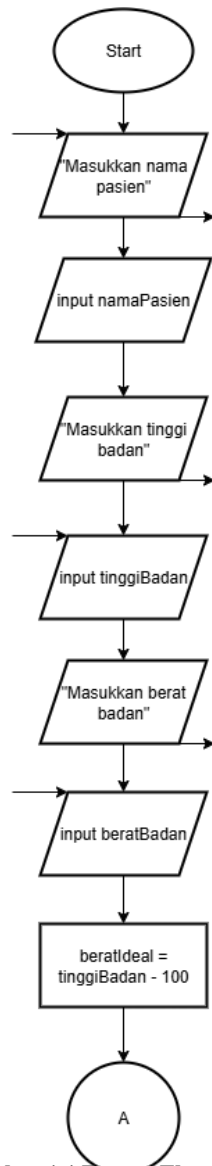
**LAPORAN PRAKTIKUM**  
**POSTTEST 2**  
**ALGORITMA PEMROGRAMAN DASAR**



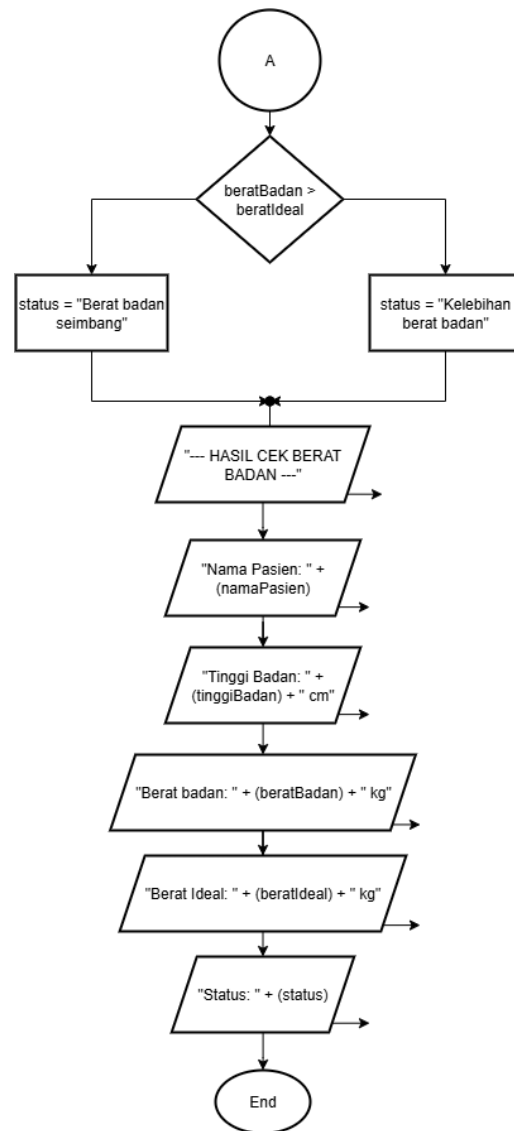
**Disusun oleh:**  
**Muhamad Radja Nur Akbar (2509106012)**  
**Kelas (A'1)**

**PROGRAM STUDI INFORMATIKA**  
**UNIVERSITAS MULAWARMAN**  
**SAMARINDA**  
**2025**

## 1. Flowchart



Gambar 1.1 Bagian Flowchart Pertama



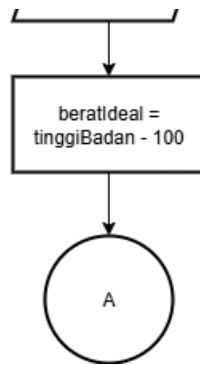
Gambar 1.2 Bagian Flowchart Kedua

## 1.1 Penjelasan



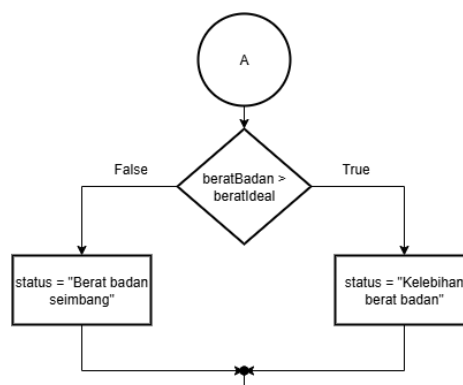
Gambar 1.3 Pecahan Satu Flowchart Pertama

Dalam **Gambar 1.3** meminta *user* untuk memasukkan variabel sesuai apa yang diminta. Terdapat 3 variabel yang akan digunakan yaitu “namaPasien” dengan tipe data *string* karena hanya meminta *user* memasukkan nama dengan huruf alfabet. Lalu ada variabel “tinggiBadan” dan “beratBadan” yang sama-sama bertipe data *float*. *Float* merupakan tipe data berupa bilangan real yang bisa menyimpan bilangan dengan desimal.



**Gambar 1.4** Pecahan Dua Flowchart Pertama

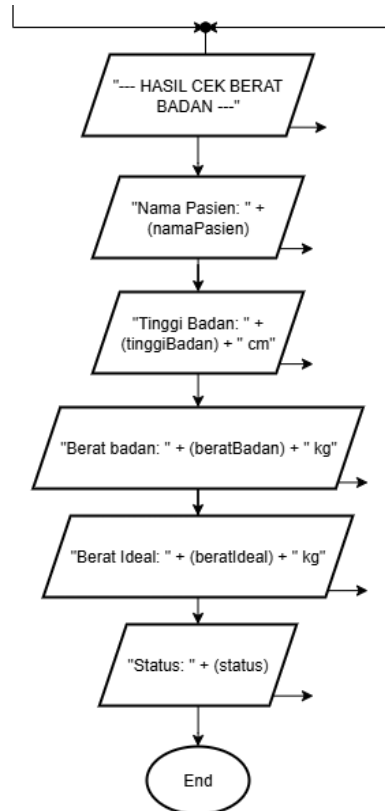
Dalam **Gambar 1.4** terdapat simbol “Process” atau “Assignment” digunakan untuk menggambarkan tugas, tindakan, atau operasi dasar dalam suatu proses. Di sini terdapat variabel baru yaitu “beratIdeal” yang diperoleh dengan mengurangi *input* variabel tinggiBadan dengan 100 maka tersimpanlah variabel beratIdeal dalam tipe data *float*. Dilanjutkan dengan simbol lingkaran dengan “A” di dalamnya, simbol ini bernama “On-Page Reference” yang berfungsi untuk menyambungkan *flowchart* yang terlalu panjang tapi masih dalam satu halaman. *Flowchart* yang dibagi harus memulai dengan simbol yang sama seperti yang ada di **Gambar 1.5**



**Gambar 1.5** Pecahan Pertama Flowchart Kedua

Di **Gambar 1.5** melanjutkan *On-Page Reference* di **Gambar 1.4**. Terdapat simbol “Decision” yang di dalamnya ada logika “>” atau lebih besar, di sini jika beratBadan lebih besar dari beratIdeal maka keluaran akan bernilai *True* dilanjutkan dengan simbol “Process” yang berisi “Kelebihan berat badan”, simbol “Process” akan menyimpan variabel status jika bernilai *True* dan saat dijalankan output dari “status” dengan tipe data *string* dan akan berisi “Kelebihan berat badan”. Begitupun sebaliknya, jika beratBadan lebih kecil beratIdeal maka

akan bernilai *False* yang berisi “Berat badan seimbang”, simbol “Process” juga menyimpan isi dari variabel status saat program dijalankan.



**Gambar 1.6** Pecahan Kedua Flowchart Kedua

**Gambar 1.6** merupakan hasil dari semua variabel, dimulai dengan *output* pembuka “--- HASIL CEK BERAT BADAN—” dan dilanjutkan dengan nama pasien, tinggi badan, berat badan, berat ideal, dan statusnya. Setiap variabel yang bertipe *float* memiliki satuan sesuai dengan yang diminta (centimeter ‘cm’ & kilogram ‘kg’).

## 2. Deskripsi Singkat Program

Program ini berfungsi untuk memberitahu apakah berat badan pasien termasuk dalam golongan ideal atau tidak berdasarkan tinggi badan dan berat badan saat ini melalui serangkaian proses yang diberikan. Manfaatnya tidak lain adalah untuk mendorong pasien untuk menjaga berat badan agar tidak menjadi penyakit seperti obesitas, diabetes, hipertensi, stroke, dan lain-lain.

## 3. Source Code

### A. Fitur Menentukan Status Berat Badan



```
#Permintaan Input kepada User dan Mendklarasikan Variabelnya
nama_pasien = input('Masukkan nama pasien: ')
tinggi_badan = float(input('Masukkan tinggi badan anda dalam cm: '))
berat_badan = float(input('Masukkan berat badan anda dalam kg: '))

#Deklarasi Variabel
berat_ideal = (tinggi_badan - 100)
is_kelebihan = berat_badan > berat_ideal
status_list = ['Berat badan seimbang', 'Kelebihan berat badan']
status = status_list[int(is_kelebihan)]

#Kode Output
print(f'''--- HASIL CEK BERAT BADAN---
Nama pasien      : {nama_pasien}
Tinggi badan     : {tinggi_badan} cm
Berat badan      : {berat_badan} kg
Berat ideal      : {berat_ideal} kg
Status           : {status}''')
```

Gambar 3.1 Kode Program

**Gambar 3.1** di atas menggunakan bahasa pemrograman Python. Program ini bertujuan untuk menstatuskan berat badan pasien termasuk ideal/seimbang atau kelebihan berat badan. Hal pertama yang dilakukan permintaan *input* kepada user yang kemudian hasil *input-an* menjadi sebuah variabel, terdapat 3 variabel yakni “nama\_pasien” bertipe *string*, “tinggi\_badan” dan “berat\_badan” yang bertipe *float*.

Selanjutnya ada variabel “berat\_ideal” yang diperoleh dengan proses aritmatika dengan mengurangi variabel “tinggi\_badan” dengan 100. Berikutnya variabel “is\_kelebihan” yang bertipe *boolean* dengan operasi logika “>”. Tipe data *boolean* merupakan tipe data yang hanya memiliki dua nilai, yakni “True” dan “False” atau benar dan salah. Salah satu operasi yang digunakan dalam boolean adalah “>” seperti pada **Gambar 3.1**. Kembali, “is\_kelebihan” akan bernilai *True* jika “berat\_badan” lebih besar dari “berat\_ideal”.

Variabel “status\_list” adalah bagian dari sintaks dalam Python yang mampu menyimpan banyak nilai dalam satu variabel. Di sini “status\_list” merupakan variabel yang menyimpan dua kemungkinan status “Berat badan seimbang” dan “Kelebihan berat badan”. Lalu variabel “status” yang bertipe *string* yang akan menampilkan salah satu dari nilai yang di dalam “status\_list” sesuai dengan kondisi yang sudah dimasukkan. Tapi kenapa ada tipe data *integer* di “status”? Seperti yang dijelaskan sebelumnya bahwa *boolean* memiliki dua nilai *True* dan *False*. Ketika menggunakan *integer* nilai dari *boolean* berubah menjadi angka *True* menjadi 1 dan *False* menjadi 0 agar bisa digunakan sebagai indeks list. Nilai dari “status\_list” akan diambil berdasarkan hasil perbandingan, di sini “Kelebihan berat badan” bernilai *True* dan “Berat badan seimbang” bernilai *false*.

Terakhir dalam **Gambar 3.1** yaitu mencetak/*print* semua variabel yang sudah dimasukkan sebelumnya, mulai dari nama pasien hingga status. Jika dilihat terdapat huruf “f” sebelum masuk ke string, ini dinamakan *f-string* yang berfungsi untuk menyematkan sebuah variabel ke dalam *string* dengan menggunakan kurung kurawal, dengan simbol ini kita tidak perlu lagi menuliskan *string* sebelum variabel yang bisa membuat program menjadi berhamburan dan tidak enak dipandang. Untuk menghindari print berulang, kita bisa menggunakan *string multiple line* ( “ ” ) jika kalimat dalam string ada banyak.

#### 4. Hasil Output

```
Masukkan nama pasien: Muhamad Radja Nur Akbar
Masukkan tinggi badan anda dalam cm: 174.5
Masukkan berat badan anda dalam kg: 64.5
--- HASIL CEK BERAT BADAN---
Nama pasien      : Muhamad Radja Nur Akbar
Tinggi badan     : 174.5 cm
Berat badan      : 64.5 kg
Berat ideal      : 74.5 kg
Status           : Berat badan seimbang
```

Gambar 4.1 Output Program Satu (Berat badan seimbang)

```
Masukkan nama pasien: Asep
Masukkan tinggi badan anda dalam cm: 160
Masukkan berat badan anda dalam kg: 85
--- HASIL CEK BERAT BADAN---
Nama pasien      : Asep
Tinggi badan     : 160.0 cm
Berat badan      : 85.0 kg
Berat ideal      : 60.0 kg
Status           : Kelebihan berat badan
```

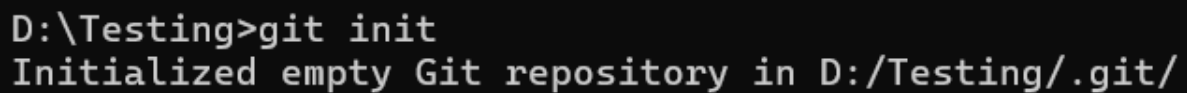
Gambar 4.2 Output Program Dua (Kelebihan berat badan)



## 5. Langkah-langkah GIT

Sebelum melakukan proses *Git*, kita harus memiliki akun GitHub dan membuat *repository* baru. Lalu siapkan *folder* yang akan di *git* dan buka Command Prompt, PowerShell, atau semacamnya untuk melakukan *git*.

### 5.1 GIT Init

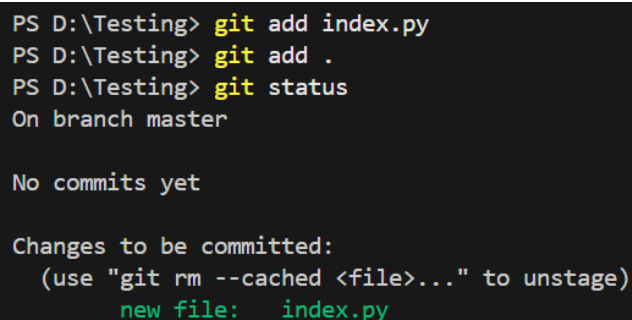


```
D:\Testing>git init
Initialized empty Git repository in D:/Testing/.git/
```

Gambar 5.1 Proses Git Init

Sebagai contoh saya akan menggunakan Command Prompt untuk melakukan *git*, langkah pertama buka *folder* yang akan di *git* lalu buka Navigation Path, ketik “cmd” untuk membuka Command Prompt di dalam *folder*, sesudah itu ketik “git init” dan *enter*. *Git init* berfungsi untuk menginisialisasi/membuat *repository git* di *folder* lokal dan memantau setiap perubahan yang terjadi di dalam *folder*.

### 5.2 GIT Add



```
PS D:\Testing> git add index.py
PS D:\Testing> git add .
PS D:\Testing> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.py
```

Gambar 5.2 Proses Git Add

*Git add* berfungsi untuk memindah semua perubahan di area kerja ke indeks, ada dua cara untuk melakukan *git add*, yang pertama dengan mengetikkan nama *file* yang ingin di add. Jika kalian ingin menambahkan banyak *file* sekaligus, ketik “.” Setelah *add* maka semua

*file* yang di dalam area kerja akan ditambahkan. Untuk mengecek apakah *file* sudah terdaftar, ketik “git status”.

### 5.3 GIT Commit

```
PS D:\Testing> git commit -m "checkpoint"
[master (root-commit) 30d681c] checkpoint
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 index.py
PS D:\Testing>
```

Gambar 5.3 Proses Git Commit

*Git commit* berfungsi untuk mengkonfirmasi setiap perubahan pada *repository* kalian dengan mengetikkan “git commit -m ‘pesan yang ingin ditulis’ ”

### 5.4 GIT Remote

```
PS D:\Testing> git remote add origin https://github.com/MRadjaNurAkbar
A25/testing.git
PS D:\Testing>
```

Gambar 5.4 Proses Git Remote

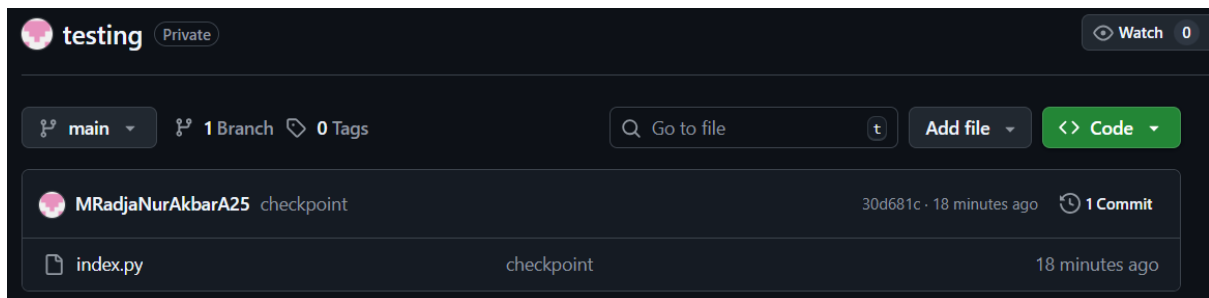
*Git remote* berfungsi untuk menghubungkan *repository* lokal ke *repository* yang ada di GitHub kalian, caranya dengan mengetik “git remote add origin ‘link repository GitHub kalian’ “

## 5.5 GIT Push

```
PS D:\Testing> git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 224 bytes | 224.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/MRadjaNurAkbarA25/testing.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Gambar 5.5 Proses Git Push

*Git push* berfungsi untuk mengunggah *file* lokal tadi ke GitHub kalian dengan mengetikkan “git push -u origin main. Kembali ke *repository* di GitHub kalian, klik *refresh* dan *file* sudah terunggah seperti pada **Gambar 5.6**



Gambar 5.6 File Lokal Sudah Terunggah di Repository GitHub