LAPORAN PRAKTIKUM

POSTTEST 6

ALGORITMA PEMROGRAMAN DASAR

Disusun oleh:

Muhamad Radja Nur Akbar (2509106012)

Kelas (A1’25)

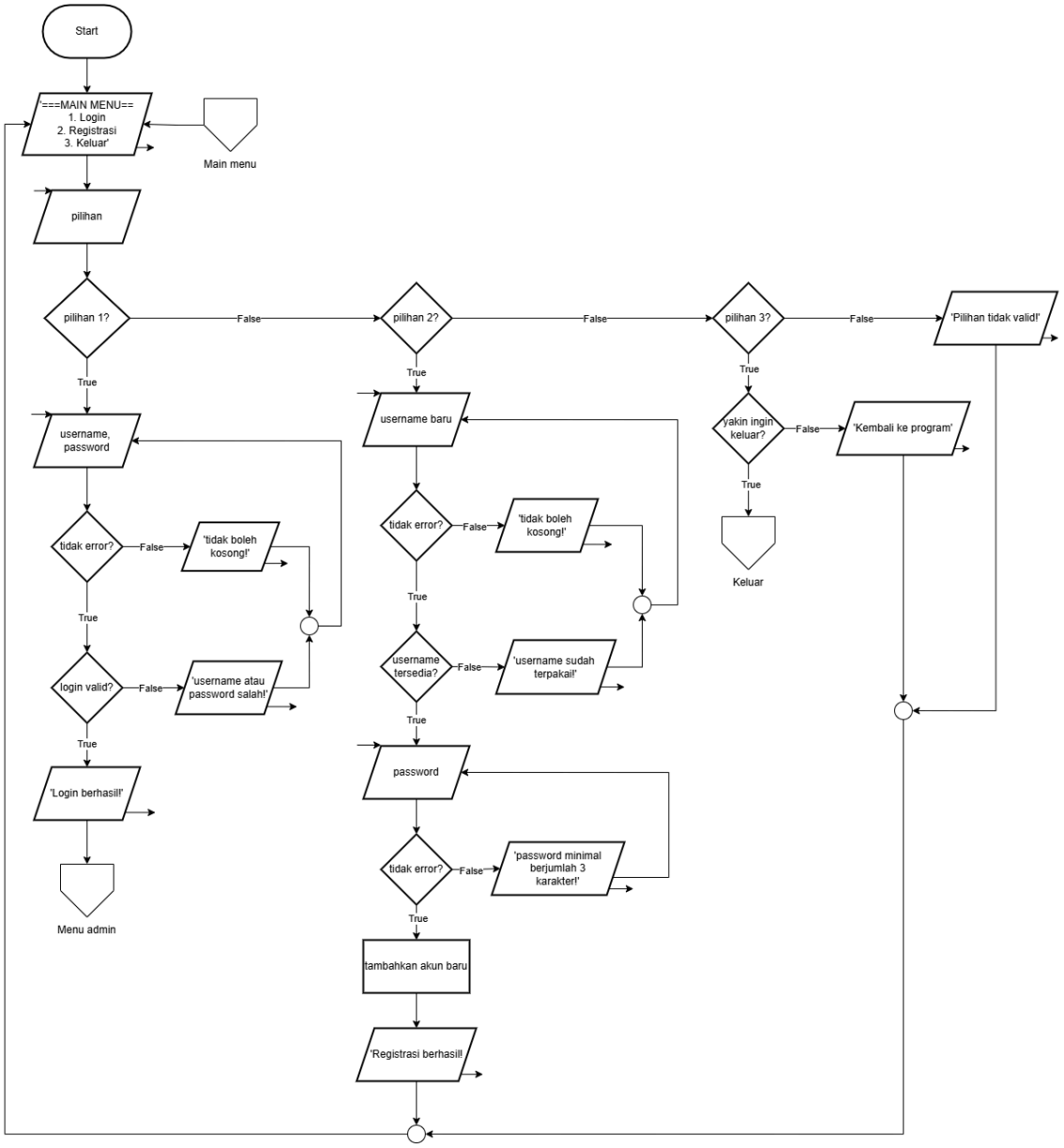
PROGRAM STUDI INFORMATIKA

UNIVERSITAS MULAWARMAN

SAMARINDA

2025

1. Flowchart

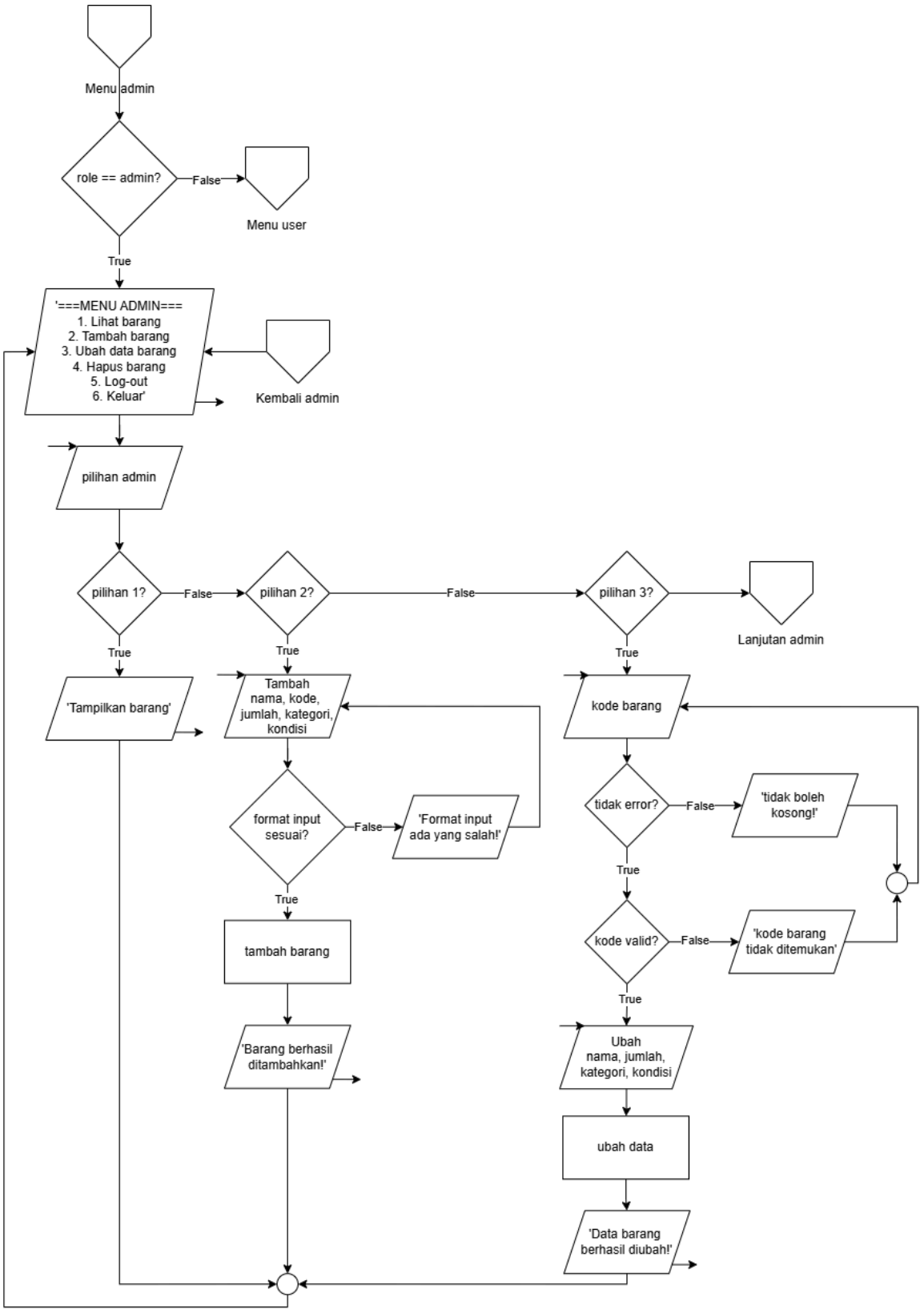


Gambar 1.1 Flowchart Main Menu

Flowchart dimulai dengan menampilkan pilihan dari *main menu*, jika pengguna memilih 1 atau *login* maka ia harus memasukkan *username* dan *password* dengan tepat. Jika kosong maka kembali *menginput* dan jika *username* dan *password* sesuai maka dilanjutkan dengan *menu admin*.

Kembali ke atas jika pengguna memilih 2 atau registrasi berarti ia membuat akun baru, setiap akun baru akan terdaftar sebagai *user*, atau *non-admin*. Proses dimulai dengan *menginput username* baru yang akan gagal jika *username* sudah digunakan. Lanjut dengan dengan membuat *password* yang minimal berjumlah 3 karakter, seperti sebelumnya *input* tidak boleh kosong. Setelah *username* dan *password* baru memenuhi persyaratan maka akun baru berhasil ditambahkan dan kembali ke *main menu*.

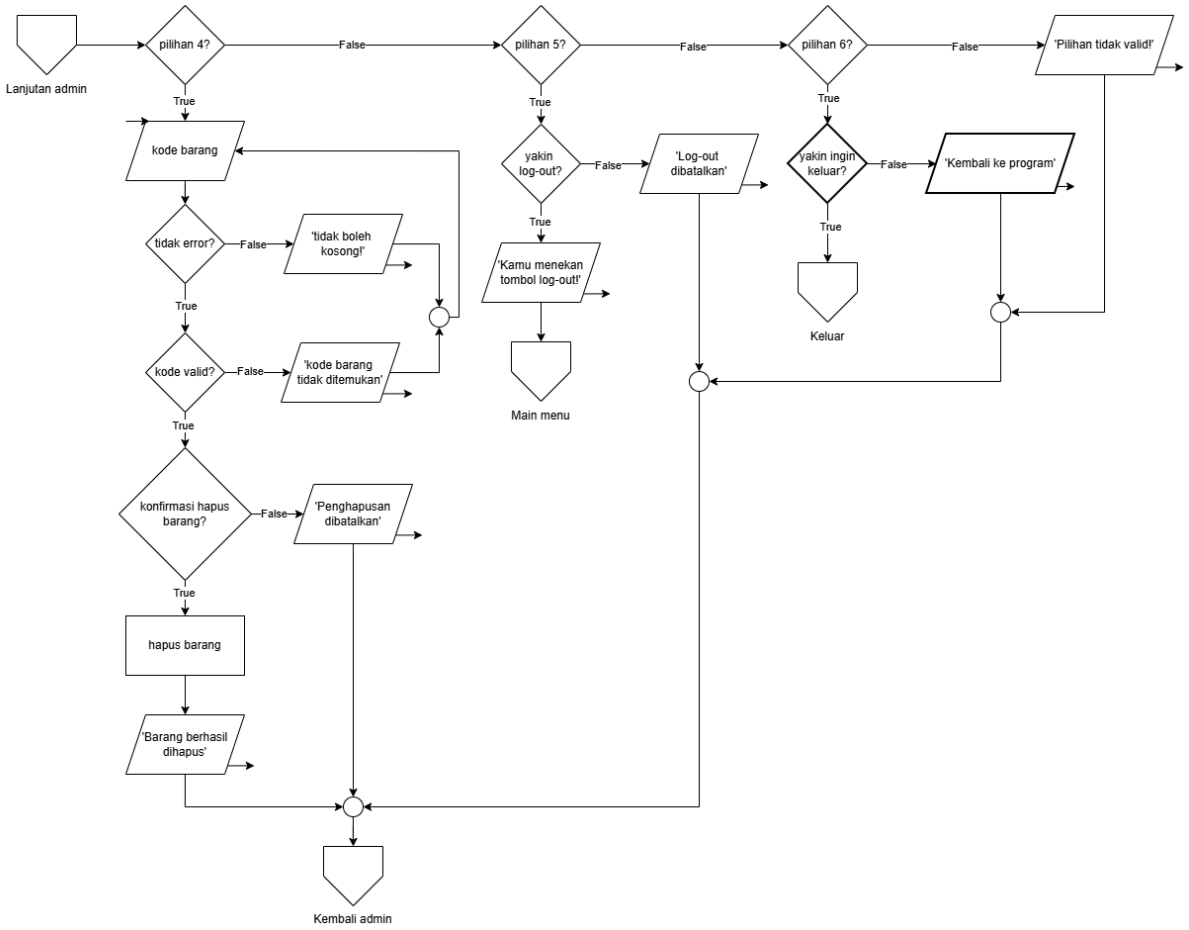
Memilih 3 atau keluar maka program akan meminta konfirmasi sebelum betul-betul keluar dari program. *Input* yang tidak valid akan membawa kembali pengguna ke *main menu*.



Gambar 1.2 Flowchart Menu admin

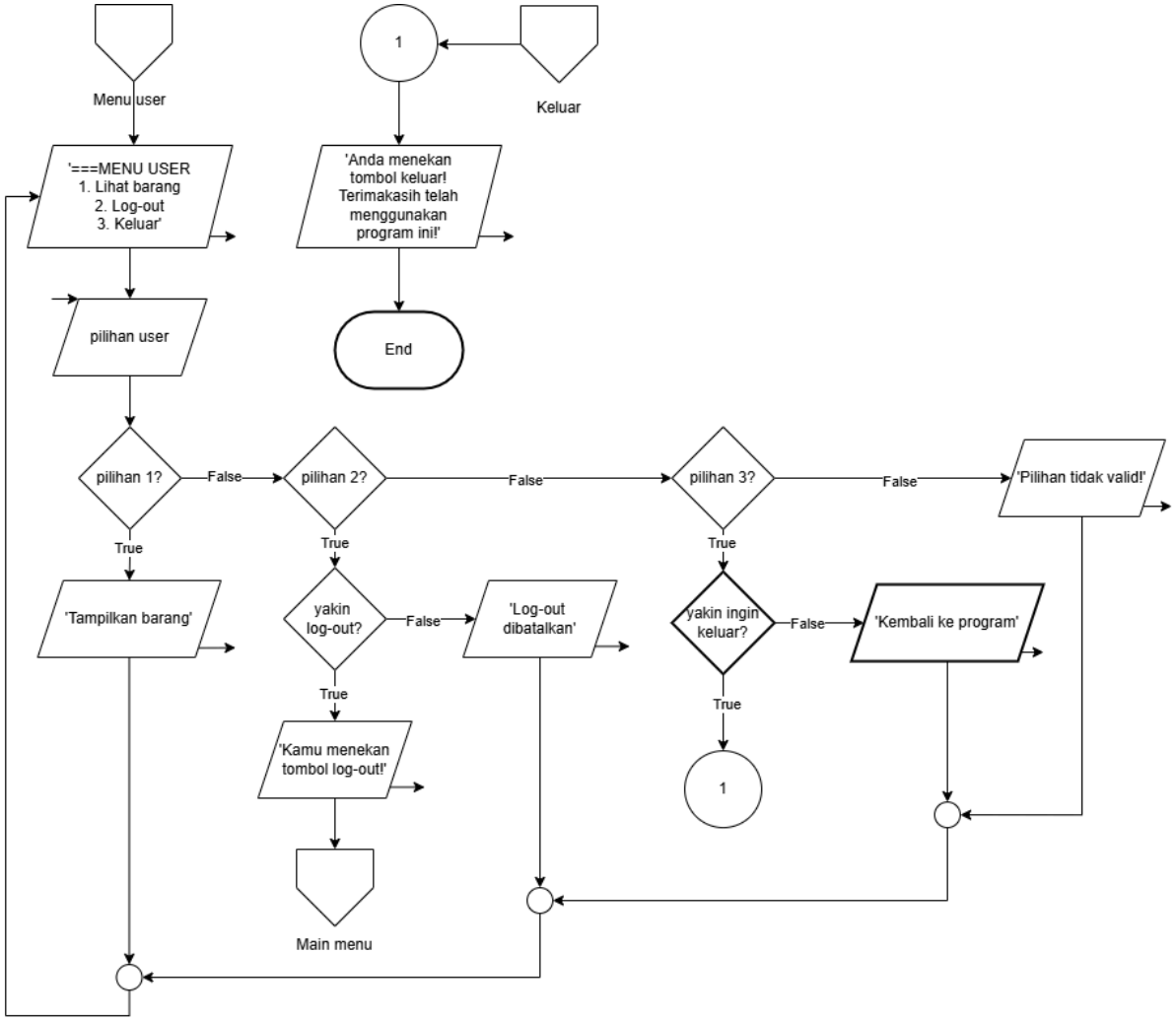
Pada *Gambar 1.2*, setelah berhasil login, *role* dari *username* akan dibaca. Seorang admin dapat melakukan 4 hal, melihat barang, menambah barang, mengubah data barang,

atau menghapus barang. Jika memilih 1 atau lihat barang, daftar barang saat ini akan ditampilkan dan kembali ke *menu* admin. Memilih 2 atau menambah barang, pengguna akan diminta memasukkan data untuk barang baru seperti nama, kode, jumlah, kategori, dan kondisi. Jika format dari *inputan* sesuai maka barang akan ditambahkan dan kembali ke *menu* admin. Jika memilih 3 atau mengubah data barang, pengguna diminta memasukkan kode barang dari salah satu barang yang ada, jika ada maka berlanjut ke mengubah data barang yang diinginkan dan perubahan akan disimpan. Jika dilihat program akan kembali ke *input* kode barang jika kode yang dimasukkan tidak terdaftar di daftar barang.



Gambar 1.3 Flowchart Lanjutan Admin

Melanjutkan *flowchart* sebelumnya, pilihan ke-4 yaitu menghapus barang, seperti sebelumnya pengguna diminta untuk memasukkan kode barang yang benar dan meminta konfirmasi sebelum betul-betul menghapus barang. Walaupun jawaban konfirmasi adalah tidak, pengguna akan dibawa kembali ke *menu* admin seperti pada *flowchart* sebelumnya. Pilihan yang lain ada *log-out* atau kembali ke *main menu* dan keluar dari program. Pilihan yang tidak valid akan kembali ke *menu admin*.



Gambar 1.4 Flowchart User dan End

Sebelumnya dijelaskan untuk setiap akun baru akan terdaftar sebagai *user*. *Login* sebagai akun baru dan *username* dan *password* benar akan membawa pengguna ke *menu user. User* hanya bisa melihat daftar barang jika memasukkan 1 pada pilihan *user*, 2 pilihan lain adalah *log-out* dan keluar seperti pada *flowchart* sebelumnya. Seluruh *off-page reference* 'Keluar' pada *flowchart* saat ini dan sebelum-sebelumnya akan membawa pengguna ke *terminator end* dan program berakhir.

2. Deskripsi Singkat Program

Program ini memberi admin pilihan dalam mengakses barang-barang kantor sementara *user* hanya bisa melihat daftar barang. Program ini juga dilengkapi fitur registrasi untuk memberi pengguna akun baru untuk digunakan.

3. Source Code

A. Dictionary

Dictionary adalah suatu tipe data pada Python yang berfungsi untuk menyimpan kumpulan data/nilai. *Dictionary* ini sendiri sesuai kalimat asalnya yaitu kamus. Dimana dalam kamus ada sebuah Kunci/Atribut dan Nilai/Informasinya. Sehingga kita cukup menggunakan kata kunci yang dimiliki oleh suatu informasi yang diinginkan untuk memanggilnya.

```
inventaris = { #Nested Dictionary
    'BRG001' : {'nama' : 'Monitor', 'jumlah' : 7, 'kategori' : 'Elektronik',
'kondisi' : 'Baik'},
    'BRG002' : {'nama' : 'Meja kayu', 'jumlah' : 10, 'kategori' : 'Perabot',
'kondisi' : 'Baik'},
    'BRG003' : {'nama' : 'Buku catatan', 'jumlah' : 30, 'kategori' : 'Alat
tulis', 'kondisi' : 'Baik'},
    'BRG004' : {'nama' : 'Kursi plastik', 'jumlah' : 20, 'kategori' :
'Perabot', 'kondisi' : 'Baik'},
    'BRG005' : {'nama' : 'Printer', 'jumlah' : 5, 'kategori' : 'Elektronik',
'kondisi' : 'Baik'},
    'BRG006' : {'nama' : 'Pulpen', 'jumlah' : 60, 'kategori' : 'Alat tulis',
'kondisi' : 'Baik'},
    'BRG007' : {'nama' : 'PC', 'jumlah' : 7, 'kategori' : 'Elektronik',
'kondisi' : 'Baik'},
}

users = {
    'radja' : {'password' : '012', 'role' : 'admin'}
}
```

Gambar 3.1 Dictionary Inventaris dan Users

Kedua *dictionary* di atas merupakan *nested dict* atau bersarang, setelah sebuah *key* dinyatakan, *value* dari *key* tersebut berisi *dict* lagi. *Dict* ‘users’ nantinya akan berupa *nested* setelah selesai registrasi. Kedua *Dictionary* ini akan disimpan sebagai ‘kamus.py’

B. Fungsi

Fungsi merupakan blok kode yang berisi suatu program untuk melakukan tugas tertentu. Tujuannya untuk menyederhanakan sebuah program dengan membuat fungsi suatu kode yang berulang, jadi jika kita membuat sebuah program yang dimana terdapat kode yang digunakan lebih dari satu kali kita bisa membuat fungsi kode tersebut agar lebih mudah dibaca dan dirawat.

```
def ulang(): #Fungsi keluar program dengan konfirmasi
    while True:
        konfirmasi = input('Konfirmasi keluar?(y/n):')
        if konfirmasi == 'y':
            print('Keluar!')
            exit()
        elif konfirmasi == 'n':
            print('Keluar dibatalkan')
            break
        else:
            print('Input tidak valid')

ulang() #Memanggil fungsi
```

Gambar 3.2.1 Membuat dan Memanggil Fungsi

Pembuatan fungsi harus diawali dengan sintaks *def* dilanjutkan dengan nama fungsi dan buka-tutup kurung. Setelah itu tekan *enter* dan maju satu *indent* dan kalian bisa membuat kode apa yang ada di fungsi tersebut. Contoh di atas merupakan fungsi untuk keluar dari sebuah program.

Sebuah fungsi bisa dipanggil dengan mengetikkan nama fungsi tersebut dan buka-tutup kurung jika fungsi tersebut berada dalam satu *file*. Namun jika berbeda, kalian harus mengimpor *file* fungsi ke *file* utama, kalian juga bisa menggunakan nama yang berbeda dari *file* fungsi tersebut dengan menggunakan *as*.

```
import cth as hei
hei.ulang()
```

```
import cth
cth.ulang()
```

Gambar 3.2.2 Memanggil Fungsi yang Berbeda File

File dari fungsi disimpan sebagai ‘cth.py’ dan untuk memanggilnya kalian bisa menggunakan satu dari dua contoh di atas. **Catatan;** suatu fungsi yang dipanggil harus terus berakhir dengan buka-tutup kurung ‘( )’ agar bisa dijalankan.

```
import os
import time
from prettytable import PrettyTable
from kamus import inventaris, users
```

Gambar 3.2.3 Impor File dan Library

Dua impor pertama merupakan *library* bawaan dari Python. *Import os* memungkinkan kita berinteraksi dengan sistem operasi dan *import time* untuk mengatur waktu, *delay* dan *timestamp*. Lalu ada *prettytable* yang merupakan *library* eksternal yang harus diinstal terlebih dahulu, ini berfungsi untuk menampilkan teks dengan *boarder*, *header*, dan perataan yang bagus. Yang terakhir merupakan import *file* dari *dictionary* yang sebelumnya disimpan sebagai 'kamus.py', karena ini tidak berisi fungsi, kita mengimpor kedua variabel *dict* yang ada di dalamnya.

```
def clear():
    os.system('cls || clear')

def delay():
    time.sleep(1)
```

Gambar 3.2.4 Fungsi Clear dan Delay

Ini merupakan contoh fungsi yang akan digunakan dalam *file* utama, *def clear* merupakan fungsi untuk membersihkan terminal dan *def delay* untuk memberi jeda 1 detik sebelum mengeksekusi kode berikutnya.

```
def daftar():
    tabel = PrettyTable ()
    tabel.field_names = ['Kode', 'Nama barang', 'Jumlah', 'Kategori',
'Kondisi']
    print('Daftar barang')
    tabel.clear_rows()
    for item, data in inventaris.items():
        tabel.add_row([item, data['nama'], data['jumlah'], data['kategori'],
data['kondisi'] ])
    return tabel
```

Gambar 3.2.5 Fungsi Tabel

Fungsi berikutnya berhubungan dengan *prettytable*, kita menyimpan 'tabel' sebagai variabel yang merupakan *PrettyTable*, lalu 'tabel.field_names' untuk menentukan *header*

setiap kolom sesuai dengan yang ditulis, dan ‘clear_rows’ berfungsi untuk menghapus objek yang sudah ada di dalam tabel jika fungsi akan digunakan secara berulang.

Berikutnya ada *for* untuk melakukan iterasi pada ‘inventaris’ dengan menggunakan ‘item’ sebagai kode barang dan ‘data’ sebagai detail barang, setiap barang yang di *dict* nantinya akan dibuat garis baru pada tabel dengan menggunakan ‘add_row’.

Terakhir ada *return*, yaitu sintaks Python untuk mengembalikan nilai/hasil dari suatu fungsi ke tempat ia dipanggil, ini sangat penting karena barangkali nilai tersebut akan dipakai dalam proses berikutnya. Dengan ini tabel akan muncul saat fungsi ‘daftar’ dipanggil.

```
def input_kosong(pesan):
    while True:
        fakta = input(pesan)
        if fakta.strip() != '':
            return fakta
        else:
            print('Input tidak boleh kosong')
```

Gambar 3.2.6 Fungsi Error Handling Inputan Kosong

Fungsi berikutnya yaitu fungsi *error handling* untuk mencegah sebuah inputan tidak memiliki nilai, disini kita menggunakan yang namanya parameter, yaitu variabel yang menampung nilai untuk diproses di dalam fungsi, ‘pesan’ disini merupakan sebuah parameter dan nantinya pesan yang di dalamnya disebut sebagai argumen yaitu nilai yang dikirim saat fungsi dipanggil.

Karena ini *error handling* maka kita menggunakan *while true*, ‘pesan’ disini fleksibel yang bisa diisi dengan pesan apa saja tergantung konteks. *Strip* yaitu *method* yang menghapus spasi yang ada di awal dan akhir sebuah *str*, jika tidak sama dengan input kosong, maka nilai ‘fakta’ akan dikembalikan, jika tidak proses akan diulang hingga memenuhi syarat.

```
def keluar():
    os.system('cls || clear')
    while True:
        konfirmasi = input('Yakin ingin keluar dari program?(y/n):')

        if konfirmasi.lower() == 'y':
            os.system('cls || clear')
```

```
            print('Anda menekan tombol keluar! Terimakasih telah menggunakan
program!')
            exit()
        elif konfirmasi.lower() == 'n':
            print('Kembali ke program')
            time.sleep(1)
            os.system('cls || clear')
            return
        else:
            print('Input tidak valid!')
```

Gambar 3.2.7 Fungsi Keluar Program

Terakhir ada fungsi untuk keluar program, mirip seperti yang dijelaskan sebelumnya, fungsi ini meminta konfirmasi sebelum keluar dari program, *method lower* digunakan untuk mengubah semua huruf *uppercase* (besar) menjadi huruf *lowercase* (kecil). Jika pengguna *menginput* 'y' maka program berakhir.

Semua fungsi ini disimpan dalam satu *file* bernama 'necfunc.py'

C. Impor

```
import os, time
from prettytable import PrettyTable
from kamus import inventaris, users
import necfunc as sip
```

Gambar 3.3.1 Impor

Sebelum memulai program utama, kita harus mengimpor kedua *file* sebelumnya, disini *file* 'necfunc.py' akan dialiaskan sebagai 'sip'.

D. Main Menu

a. Login

```
while True: #Loop utama program
    role = None
    loginn = False
    while True:
        print('''===Menu===
1. Login
2. Register
3. Keluar''')
        menu = input('Pilih menu:')

        if menu == '1': #Menu login
```

```
            while True:
                print('Login akun')
                usr = sip.input_kosong('Masukkan Username\t:')
                pw = sip.input_kosong('Masukkan password\t:')

                for u in users:
                    if usr == u and pw == users[u]['password']:
                        role = users[u]['role']
                        loginn = True
                        break
                if loginn:
                    sip.clear()
                    print('Login berhasil!')
                    sip.delay()
                    break
                else:
                    sip.clear()
                    print('Username atau password salah!')
            if loginn:
                break
```

Gambar 3.4.1 Proses Login

Pilihan pertama dari *main menu* adalah *login*, *input* dari ‘usr’ dan ‘pw’ akan menggunakan fungsi ‘input_kosong’ untuk mencegah *input* kosong.

Setelah itu akan ada pengecekkan *username* dan *password*, pada *dict* ‘users’, ‘u’ merupakan *username* atau *key* yang ada di *dict* tersebut, lalu kemudian *password* akan dicek dengan melihat *value* dari ‘u’ yang sesuai, jika *password* sesuai dengan yang ada di *dict* ‘users’ maka *role* akan menjadi ‘role’ yang juga ada di dalam *dict* dan status ‘loginn’ yang awalnya *false* menjadi *true*.

Jika ‘loginn’ *true*, maka proses *login* berhasil dan lanjut ke *menu* sesuai *rolenya*. Dua *break* untuk mengakhir *loop login* dan *loop main menu*.

b. Registrasi

```
elif menu == '2':
            sip.clear()
            print('Registrasi')

            while True:
                cek_ada_usr = False
                usr_baru = sip.input_kosong('Masukkan Username baru\t:')

                for u in users:
                    if usr_baru == u:
```

```
                cek_ada_usr = True
                break
```

Gambar 3.4.2 Input Username Baru

Pilihan kedua adalah registrasi atau membuat akun baru. Proses awal yaitu pencegahan *username* baru sama dengan *username* yang sudah ada. Seperti sebelumnya 'u' berisi *key* dari *dict* 'users', jika ada yang sama makan nilai 'cek_ada_usr' menjadi *true* dan perulangan di-*break*.

```
            if cek_ada_usr:
                print('Username sudah terpakai! Gunakan Username lain')
            else:
                while True:
                    pw_baru = input('Masukkan password\t:')
                    if len(pw_baru) < 3:
                        print('Password minimal berjumlah 3 karakter!')
                    else:
                        break
                users.update({
                    usr_baru : {'password' : pw_baru,
                                'role' : 'user'
                    }
                })
                sip.clear()
                print('Registrasi berhasil!')
                break
```

Gambar 2.4.3 Pengecekkan Username dan Input Password

Program akan kembali ke proses *input username* jika *username* baru telah digunakan. Jika tidak lanjut ke *input password*. *Password* disini memiliki syarat jika kurang dari 3 karakter maka akan *input* ulang. Jika *username* dan *password* sudah tepat, *key* baru akan ditambahkan ke *dict* 'users' dengan menggunakan *update*, karena ini *nested*, *key* baru memiliki *value* yaitu 'pw_baru' dan 'user'. Dalam program ini semua akun baru memiliki *role user* biasa.

```
        elif menu == '3':
            sip.keluar()

        else:
            os.system('cls || clear')
            print('Input tidak valid!')
```

Gambar 3.4.5 Keluar dan Input Tidak Valid

Disini penerapan dari fungsi ‘keluar’ kita hanya perlu mengetikkan ‘sip’ diikuti dengan nama fungsi dan buka-tutup kurung daripada menuliskan semua kode yang bisa memakan hingga 15 *line*. Dan *else* yang merupakan *error handling* jika *input* di luar pilihan.

E. Menu Admin

```
while True:
        if role == 'admin': #Menu admin
            print(f'Halo, {usr}!')
            print('''===MENU ADMIN===
1. Lihat barang
2. Tambah barang
3. Ubah data barang
4. Hapus barang
5. Log-out
6. Keluar''')
            menu_admin = input('Pilih menu:')
```

Gambar 3.5 Menu Admin

Dalam *menu* admin, ada enam pilihan, setelah memilih salah satu, program akan masuk ke pilihan tersebut/

a. Lihat Barang

```
if menu_admin == '1':
                sip.clear()
                print(sip.daftar())
```

Gambar 3.5.1 Menu Lihat Barang

Pilihan ini hanya sekadar melihat daftar barang dalam bentuk tabel dengan memanggil fungsi ‘daftar’ atau *prettytable* seperti pada sub-bab sebelumnya.

b. Tambah Barang

```
elif menu_admin == '2':

                sip.clear()
                print('Tambah barang')

                nama_barang = sip.input_kosong('Nama barang\t:')

                while True:
                    cek_kode_barang = False
                    kode_barang = sip.input_kosong('Kode barang\t:')

                    for j in inventaris:
                        if kode_barang == j:
                            cek_kode_barang = True
                            break
                    if cek_kode_barang:
                        print('Kode barang sudah terpakai! Gunakan kode
lain!')
                    else:
                        break
```

Gambar 3.5.2 Proses Membuat Nama Barang dan Kode

Setelah mengisi nama barang baru, kode barang juga akan diminta, karena kode sangat esensial yang menjadi identitas utama, maka akan dilakukan pengecekkan. Iterasi dilakukan kembali dengan adanya *for*, setiap *key* pada 'inventaris' akan dicek, jika ada yang sama dengan kode barang baru, maka harus *menginput* kode lain.

```
while True:
                        jumlah_input = input('Jumlah barang\t:')
                        if jumlah_input.isdigit() and int(jumlah_input) > 0:
                            jumlah_barang = int(jumlah_input)
                            break
                        else:
                            print('Jumlah harus tidak boleh kosong dan harus
berupa angka lebih dari 0!')
```

Gambar 3.5.3 Menambahkan Jumlah Barang

Berikutnya jumlah, jumlah barang harus lebih dari nol dan berupa angka, lalu variabel 'jumlah_barang' berisi nilai yang sama dengan *input* dari 'jumlah_input'.

```
kategori_barang = sip.input_kosong('Kategori barang\t:')

                kondisi_barang = sip.input_kosong('Kondisi barang\t:')

                inventaris.update({
                    kode_barang : {'nama' : nama_barang,
                                'jumlah' : jumlah_barang,
                                'kategori' : kategori_barang,
                                'kondisi' : kondisi_barang}
                })
                sip.clear()
                print('Barang berhasil ditambahkan!')
```

Gambar 3.5.4 Menambahkan Barang

Setelahnya ada kategori dan kondisi barang dengan fungsi yang sama lalu menambahkan barang baru tersebut ke *dict* 'inventaris' menggunakan *update* dan format yang sama.

c. Ubah Data Barang

```
elif menu_admin == '3':
                ditemukan = False
                sip.clear()
                print('Ubah data barang')
                print(sip.daftar())

                cari_kode = sip.input_kosong('Masukkan kode barang yang
ingin diubah\t:')
                sip.clear()


                for k in inventaris:
                    if k == cari_kode:
                        ditemukan = True
                        ada = inventaris[k]
```

Gambar 3.5.5 Pengecekkan Kode Barang

Selain melihat dan menambah barang, admin juga bisa mengubah data barang yang sudah ada, pertama pengguna harus menginput kode barang yang ingin diubah datanya, setelah itu kode barang akan dicari di 'inventaris' sebagai 'k', jika ada yang sama maka variabel 'ditemukan' menjadi *true* dan 'ada' yang berisi *key* dari 'k'.

```
print(f'''
    Data ditemukan:
    Nama\t\t: {ada['nama']}
    Kode\t\t: {k}
    Jumlah\t\t: {ada['jumlah']}
    Kategori\t: {ada['kategori']}
    Kondisi\t\t: {ada['kondisi']}''')

                        print('''Pilih data yang ingin diubah:
    1. Ubah nama barang
    2. Ubah jumlah barang
    3. Ubah kategori
    4. Ubah kondisi''')
                        pilihan_ubah = input('Pilih apa yang ingin diubah:
')
```

Gambar 3.5.6 Data Barang dan Pilihan Ubah

Setelah kode barang sesuai, data atau *field* dari kode barang akan ditampilkan beserta opsi apa saja yang bisa diubah.

```
if pilihan_ubah == '1':
                            nama_baru = sip.input_kosong('Ubah nama
barang\t:')
                            inventaris[k]['nama'] = nama_baru
                            sip.clear()
                            print('Nama barang berhasil diubah')
                            break
```

Gambar 3.5.7 Ubah Nama Barang

Susunan ini dimulai dengan mengakses ‘inventaris’, karena sebelumnya ‘k’ itu sama dengan ‘cari_kode’ maka cukup menggunakan ‘k’ saja, lalu salah satu *value* yang ada di ‘k’ yaitu ‘nama’. Maka nama akan diubah.

```
elif pilihan_ubah == '2':
                            sip.clear()
                            print(f"Jumlah barang saat ini :
{ada['jumlah']}")
                            print('''Pilih:
    1. Tambah jumlah barang
    2. Kurangi jumlah barang''')

                            while True:
                                tamkur = input('Pilih (1/2): ')
                                if tamkur in ['1', '2']:
                                    break
                                else:
```

```
                                    print('Pilihan tidak valid!')
```

Gambar 3.5.8 Ubah Jumlah Barang

Jumlah barang saat ini akan ditampilkan sebelum memilih pilihan menambah atau mengurangi barang, ‘tamkur’ menggunakan struktur percabangan yang berbeda tanpa menggunakan *elif*, disini jika *input* dari ‘tamkur’ adalah ‘1’ atau ‘2’ maka *while true* ‘tamkur’ diakhiri.

```
while True:
                                jumlah_tamkur = input('Masukkan jumlah: ')
                                if jumlah_tamkur.isdigit() and
int(jumlah_tamkur) > 0:
                                    proses_operasional = int(jumlah_tamkur)

                                    if tamkur == '1':
                                        ada['jumlah'] += proses_operasional
                                        sip.clear()
                                        print('Jumlah barang berhasil
ditambahkan!')
                                        break

                                    elif tamkur == '2':
                                        if ada['jumlah'] >=
proses_operasional:
                                            ada['jumlah'] -=
proses_operasional
                                            sip.clear()
                                            print('Jumlah barang berhasil
dikurangi!')
                                            break
                                        else:
                                            print('Jumlah terlalu besar
untuk dikurangi')
                                else:
                                    print('Harus berupa angka lebih dari
0!')
```

Gambar 2.5.9 Proses Operasional

Setelah memilih operasional, permintaan berikutnya yaitu besar jumlah yang ingin dioperasikan, ini menggunakan *nested if* sebelum *else* yang artinya *if* kedua akan jalan jika *if* sebelumnya terpenuhi, disini besar jumlah harus berupa angka lebih dari nol.

Bergantung pada pilihan ‘tamkur’ sebelumnya, jika menambahkan maka jumlah barang saat ini ditambah dengan besar jumlah. Jika mengurangi maka ada syarat lagi, jika

besar jumlah melebihi jumlah barang saat ini, maka akan dilakukan *input* ulang hingga jumlah barang berhasil dikurangi.

```
elif pilihan_ubah == '3':
                            kategori_baru  = sip.input_kosong('Ubah kategori
barang\t:')
                            inventaris[k]['kategori'] = kategori_baru
                            sip.clear()
                            print('Kategori barang berhasil diubah')
                            break

                        elif pilihan_ubah == '4':
                            kondisi_baru  = sip.input_kosong('Ubah kondisi
barang\t:')
                            inventaris[k]['kondisi'] = kondisi_baru
                            print('Kondisi barang berhasil diubah')
                            break
                if not ditemukan:
                    sip.clear()
                    print('Kode barang tidak ditemukan')
```

Gambar 3.5.10 Ubah Kategori dan Kondisi Barang

Proses mengubah kategori dan kondisi barang sama saja saat mengubah nama barang. Sebelumnya jika kode barang sesuai nilai ‘ditemukan’ menjadi *true*, jika tidak sesuai maka kembali ke *menu* admin. Setiap perubahan yang berhasil juga membawa pengguna kembali ke *menu* admin.

d. Hapus Barang

```
elif menu_admin == '4':
                sip.clear()
                print('Hapus barang')

                while True:
                    ketemu = False
                    print(sip.daftar())

                    cari_kode_hapus = sip.input_kosong('Masukkan kode barang
yang ingin dihapus\t:')

                    for l in inventaris:
                        if l == cari_kode_hapus:
                            ketemu = True
                            ama = inventaris[l]
```

```
                            print(f'''
    Data ditemukan:
    Nama\t\t: {ama['nama']}
    Kode\t\t: {l}
    Jumlah\t\t: {ama['jumlah']}
    Kategori\t: {ama['kategori']}
    Kondisi\t\t: {ama['kondisi']}''')
                            hapus = input('Yakin ingin menghapus
barang?(y/n)')

                            if hapus.lower() == 'y':
                                inventaris.pop(l)
                                sip.clear()
                                print('Barang berhasil dihapus!')
                                break
                            elif hapus.lower() == 'n':
                                print('Penghapusan dibatalkan')
                                break
                            else:
                                print('Input tidak valid!')
                                break
```

Gambar 3.5.11 Cek Kode Barang yang Akan Dihapus

Sama seperti sebelumnya, program meminta *menginput* kode barang yang kemudian kode barang yang sesuai akan dicari hingga ketemu, 'ketemu' menjadi *true* jika sesuai. Selanjutnya ada konfirmasi, ketiga pilihan sama-sama membawa pengguna kembali ke *menu* admin jika proses sudah selesai. Jika *menginput* 'y' maka barang akan dihapus.

e. Log-out dan Keluar

```
elif menu_admin == '5':
                sip.clear()
                kembali1 = input('Yakin ingin log-out?(y/n): ')
                if kembali1.lower() == 'y':
                    os.system('cls || clear')
                    print('Kamu menekan tombol log-out!')
                    break
                elif kembali1.lower() == 'n':
                    print('Log-out dibatalkan')
                else:
                    print('Pilihan tidak valid!')

            elif menu_admin == '6':
                sip.keluar()
```

```
            else:
                sip.clear()
                print('Input tidak valid!')
```

Gambar 3.5.12 Proses Log-Out dan Keluar

Proses *loog-out* mirip seperti keluar, hanya saja hanya mengembalikan pengguna ke *main menu* jika *menginput* ‘y’. Lalu ada pilihan keluar dan *error handling* di *else* jikalau *input* ‘menu_admin’ tidak ada yang sesuai.

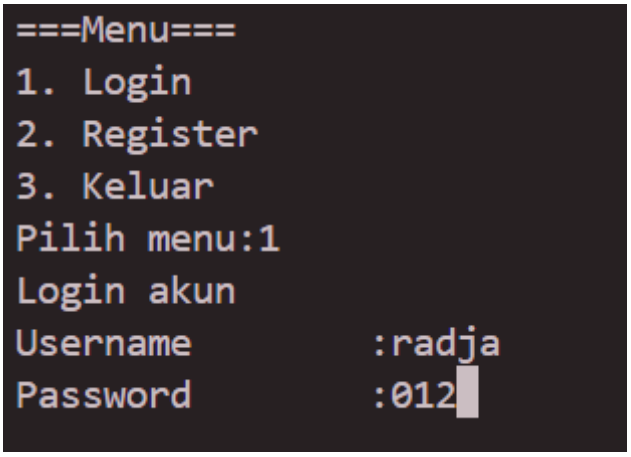
F. Menu User

```
else:
            print(f'Halo, {usr}!')
            print('''===MENU USER===
1. Lihat barang
2. Log-out
3. Keluar''')
            menu_user = input('Pilih menu: ')

            if menu_user == '1':
                sip.clear()
                print('Daftar barang')
                print(sip.daftar())

            elif menu_user == '2':
                sip.clear()
                kembali2 = input('Yakin ingin log-out?(y/n): ')
                if kembali2.lower() == 'y':
                    os.system('cls || clear')
                    print('Kamu menekan tombol log-out!')
                    break
                elif kembali2.lower() == 'n':
                    print('Log-out dibatalkan')
                else:
                    print('Pilihan tidak valid!')

            else:
                sip.keluar()
```

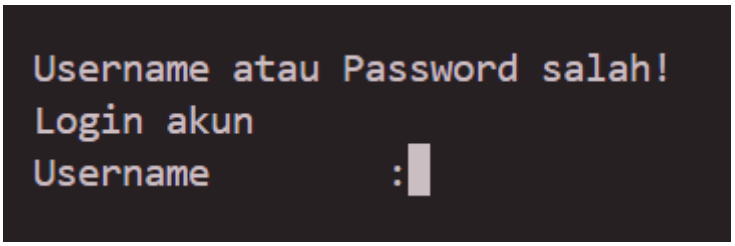
Gambar 3.6 Menu User

Jika pengguna *login* sebagai *user*, mereka hanya memiliki tiga pilihan, yaitu lihat barang, *log-out*, dan keluar dari program, dengan menggunakan kode yang sama, *menu user* secara keseluruhan sangat sederhana dan tidak rumit untuk dipahami.

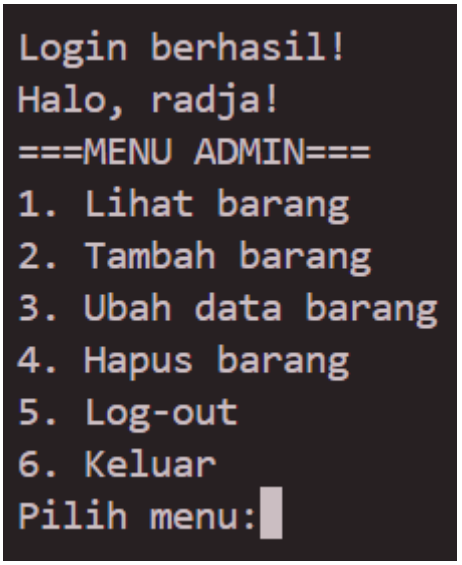
4. Hasil Output



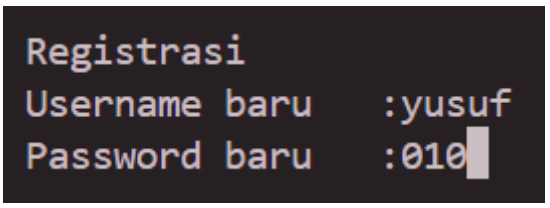
Gambar 4.1 Login



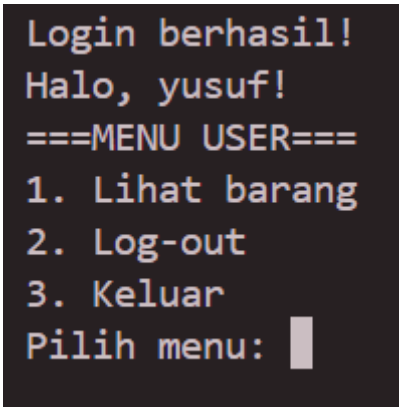
Gambar 4.2 Username atau Password Salah



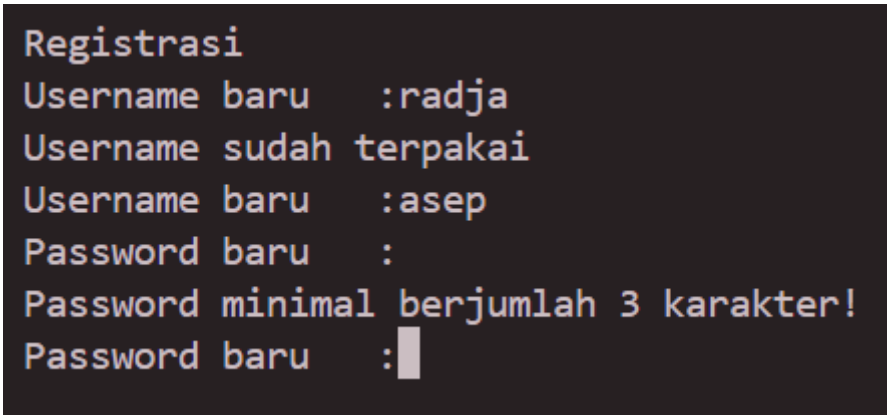
Gambar 4.3 Login Sebagai radja dan Berhasil



Gambar 4.4 Registrasi



Gambar 4.5 Login Sebagai User



Gambar 4.6 Kesalahan saat Registrasi

```
Tambah barang
Nama barang      :Keyboard
Contoh kode barang: BRG001, BRG002
Kode barang      :BRG006
Kode barang sudah digunakan! Gunakan kode lain
Contoh kode barang: BRG001, BRG002
Kode barang      :BRG008
Jumlah barang    :5
Kategori barang :Elektronik
Kondisi barang   :Baik
```

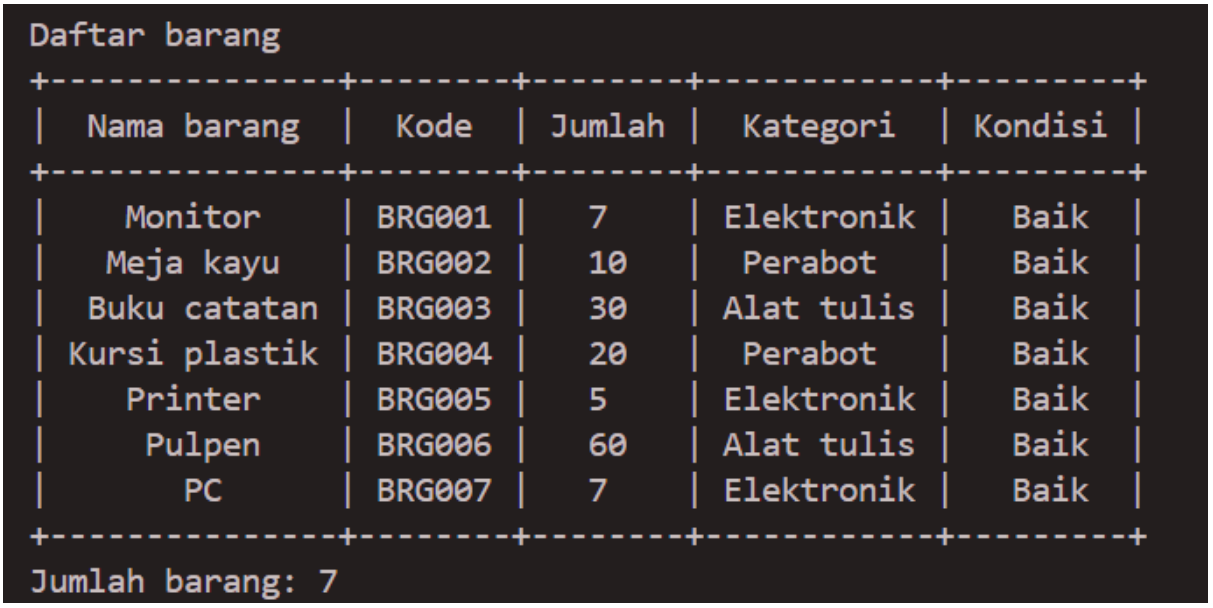
Gambar 4.7 Proses Menambahkan Barang

```
Ubah jumlah barang:
Jumlah barang saat ini: 60
Pilih:
1. Tambah jumlah barang
2. Kurangi jumlah barang
Pilih (1/2): 2
Masukkan jumlah: 10
```

Gambar 4.8 Mengubah Jumlah Barang (Pulpen)

```
Daftar barang
+---------------+--------+--------+------------+---------+
|  Nama barang  |  Kode  | Jumlah |  Kategori  | Kondisi |
+---------------+--------+--------+------------+---------+
|    Monitor    | BRG001 |   7    | Elektronik |   Baik  |
|   Meja kayu   | BRG002 |   10   |  Perabot   |   Baik  |
|  Buku catatan | BRG003 |   30   | Alat tulis |   Baik  |
| Kursi plastik | BRG004 |   20   |  Perabot   |   Baik  |
|    Printer    | BRG005 |   5    | Elektronik |   Baik  |
|     Pulpen    | BRG006 |   50   | Alat tulis |   Baik  |
|       PC      | BRG007 |   7    | Elektronik |   Baik  |
|    Keyboard   | BRG008 |   5    | Elektronik |   Baik  |
+---------------+--------+--------+------------+---------+
Hapus barang
Masukkan kode barang yang ingin dihapus: BRG005

Data ditemukan:
Nama            : Printer
Kode            : BRG005
Jumlah          : 5
Kategori        : Elektronik
Kondisi         : Baik
Konfirmasi hapus barang?(y/n):y
```

Gambar 4.9 Menghapus Barang

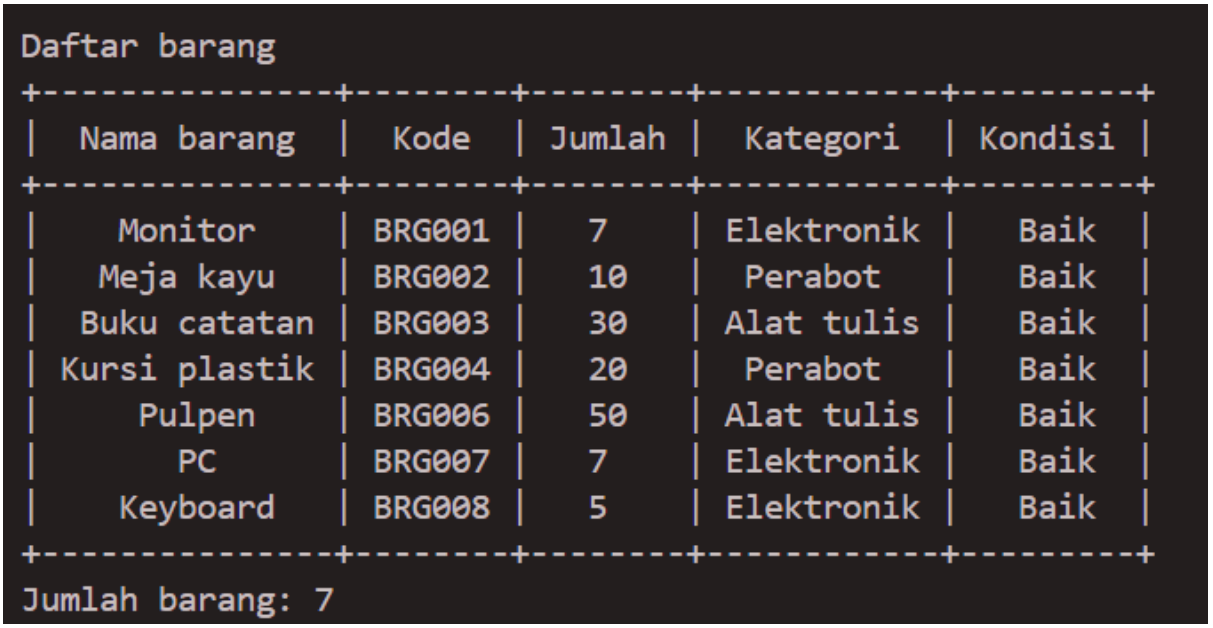


Daftar barang

Nama barang	Kode	Jumlah	Kategori	Kondisi
Monitor	BRG001	7	Elektronik	Baik
Meja kayu	BRG002	10	Perabot	Baik
Buku catatan	BRG003	30	Alat tulis	Baik
Kursi plastik	BRG004	20	Perabot	Baik
Printer	BRG005	5	Elektronik	Baik
Pulpen	BRG006	60	Alat tulis	Baik
PC	BRG007	7	Elektronik	Baik

Jumlah barang: 7

Gambar 4.10 Tabel Sebelum CRUD



Daftar barang

Nama barang	Kode	Jumlah	Kategori	Kondisi
Monitor	BRG001	7	Elektronik	Baik
Meja kayu	BRG002	10	Perabot	Baik
Buku catatan	BRG003	30	Alat tulis	Baik
Kursi plastik	BRG004	20	Perabot	Baik
Pulpen	BRG006	50	Alat tulis	Baik
PC	BRG007	7	Elektronik	Baik
Keyboard	BRG008	5	Elektronik	Baik

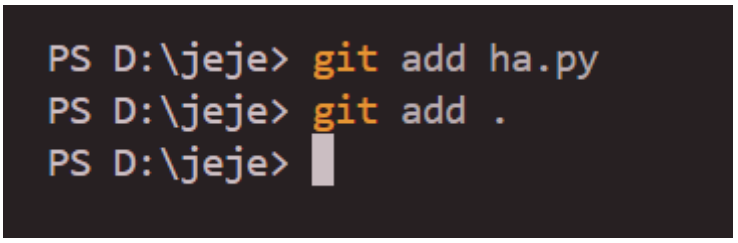
Jumlah barang: 7

Gambar 4.11 Tabel Sesudah CRUD

5. Langkah-Langkah GIT

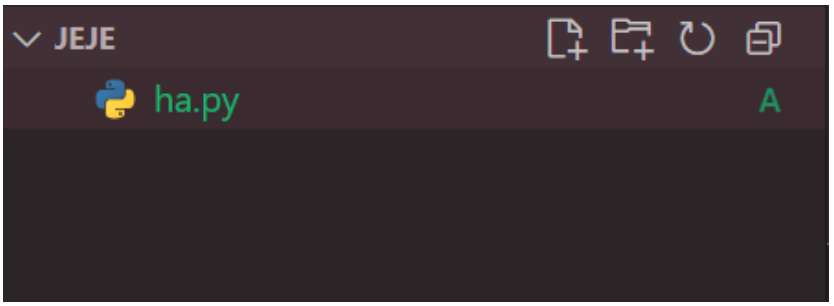
Jika dalam suatu folder kita sudah pernah melakukan *git init* dan *git remote* maka kita tidak perlu lagi melakukannya dan langsung ke *git add*.

5.1 GIT Add



Gambar 5.1 Proses Git Add

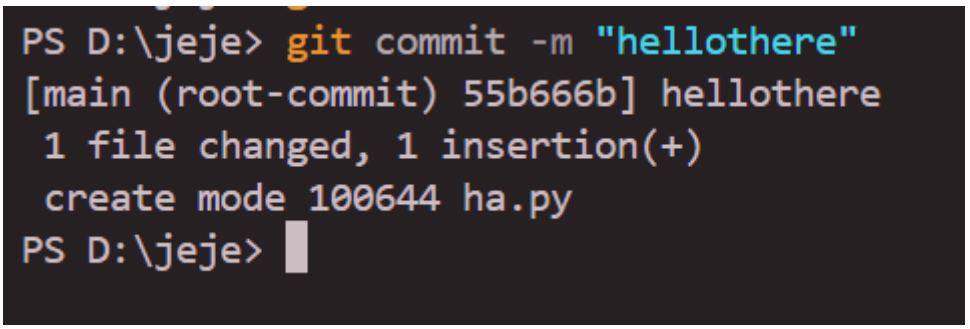
Git add berfungsi untuk memindah semua perubahan di area kerja ke indeks, ada dua cara untuk melakukan *git add*, yang pertama dengan mengetikkan nama *file* yang ingin di *add*. Jika kalian ingin menambahkan banyak *file* sekaligus, ketik "git add ."



Gambar 5.2 Sebuah File yang Telah di Add

File yang telah di *add* akan menampilkan hurus 'A' tepat di sebelah nama *file*, ini menandakan *file* sudah ditambahkan.

5.2 GIT Commit



Gambar 5.3 Proses Git Commit

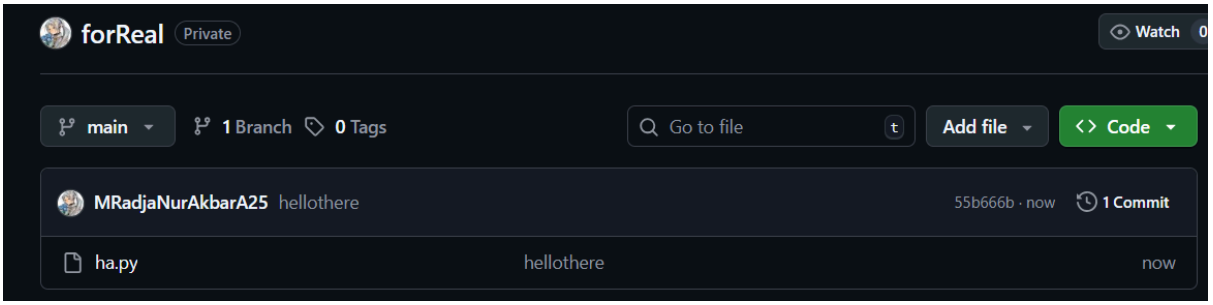
Git commit berfungsi untuk mengkonfirmasi setiap perubahan pada *repository* kalian dengan mengetikkan “git commit -m ‘pesan yang ingin ditulis’ ”

5.3 GIT Push

```
PS D:\jeje> git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 235 bytes | 235.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/MRadjaNurAkbarA25/forReal.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS D:\jeje>
```

Gambar 5.4 Repository Lokal diunggah ke Github

Git push berfungsi untuk mengunggah *file* lokal tadi ke GitHub kalian dengan mengetikkan “git push -u origin main”.



Gambar 5.5 File sudah terunggah di Akun Github