LAPORAN PRAKTIKUM

POSTTEST 7

ALGORITMA PEMROGRAMAN DASAR

Disusun oleh:

Muhamad Radja Nur Akbar (2509106012)

Kelas (A1’25)

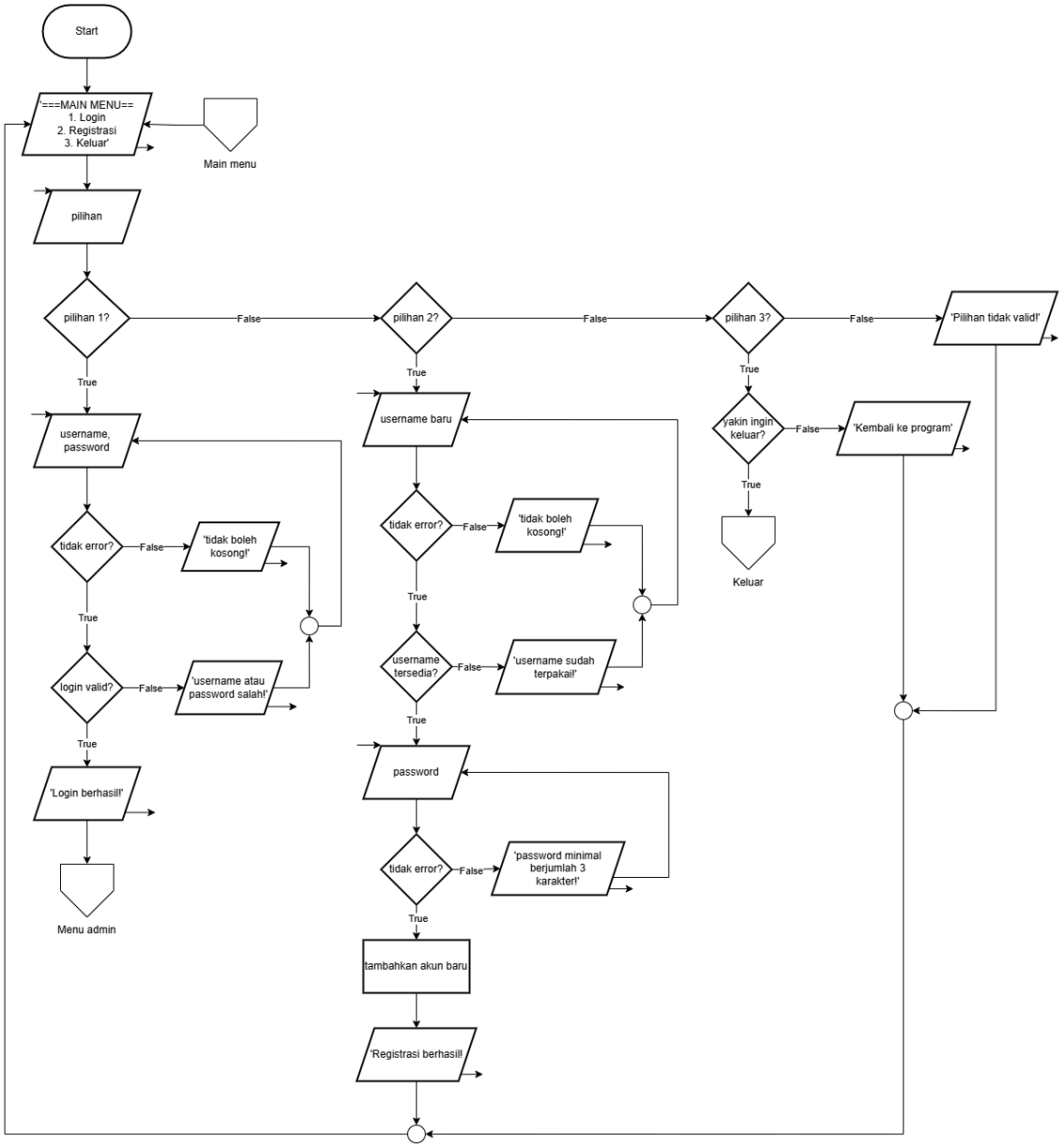
PROGRAM STUDI INFORMATIKA

UNIVERSITAS MULAWARMAN

SAMARINDA

2025

1. Flowchart

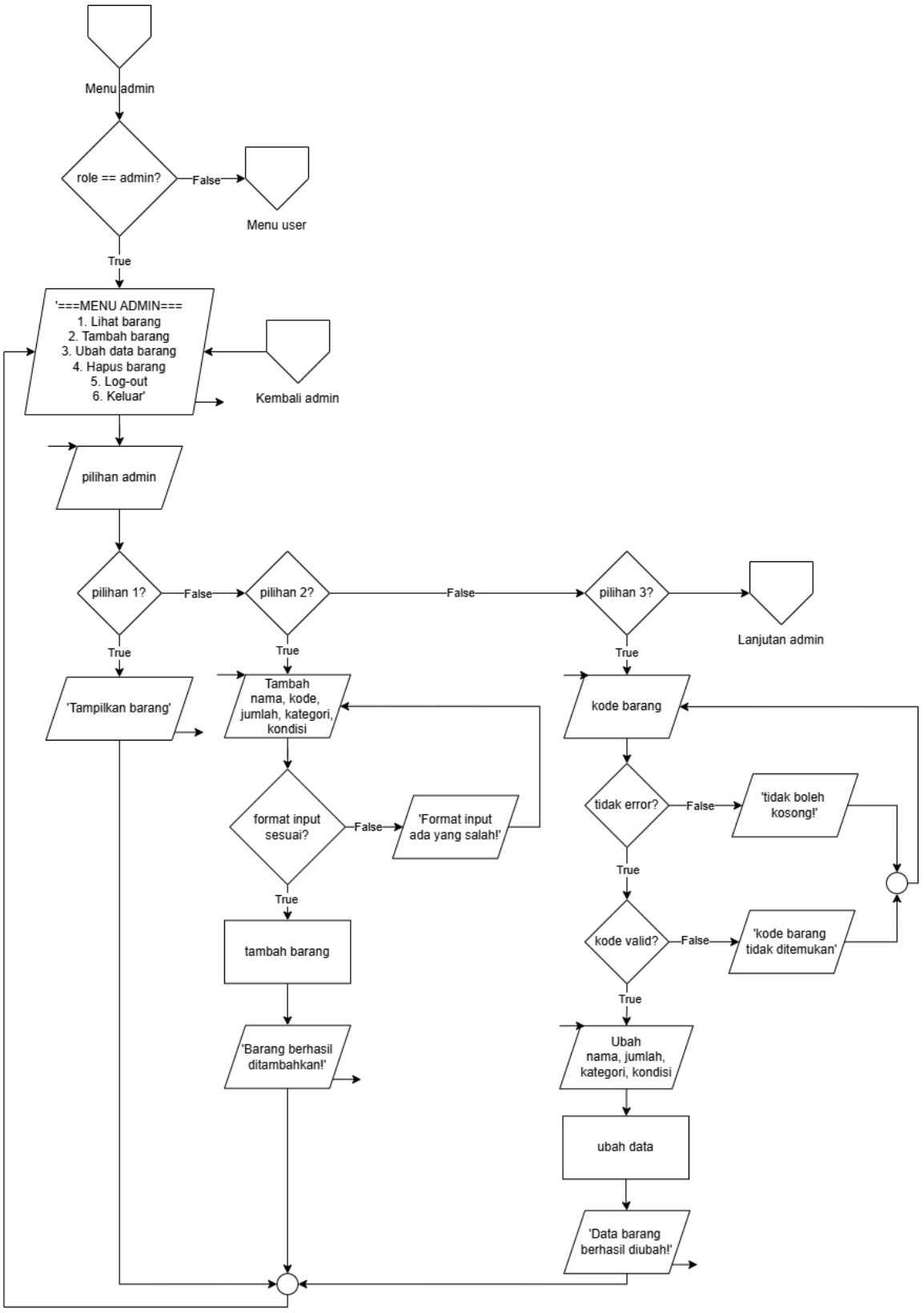


Gambar 1.1 Flowchart Main Menu

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*.

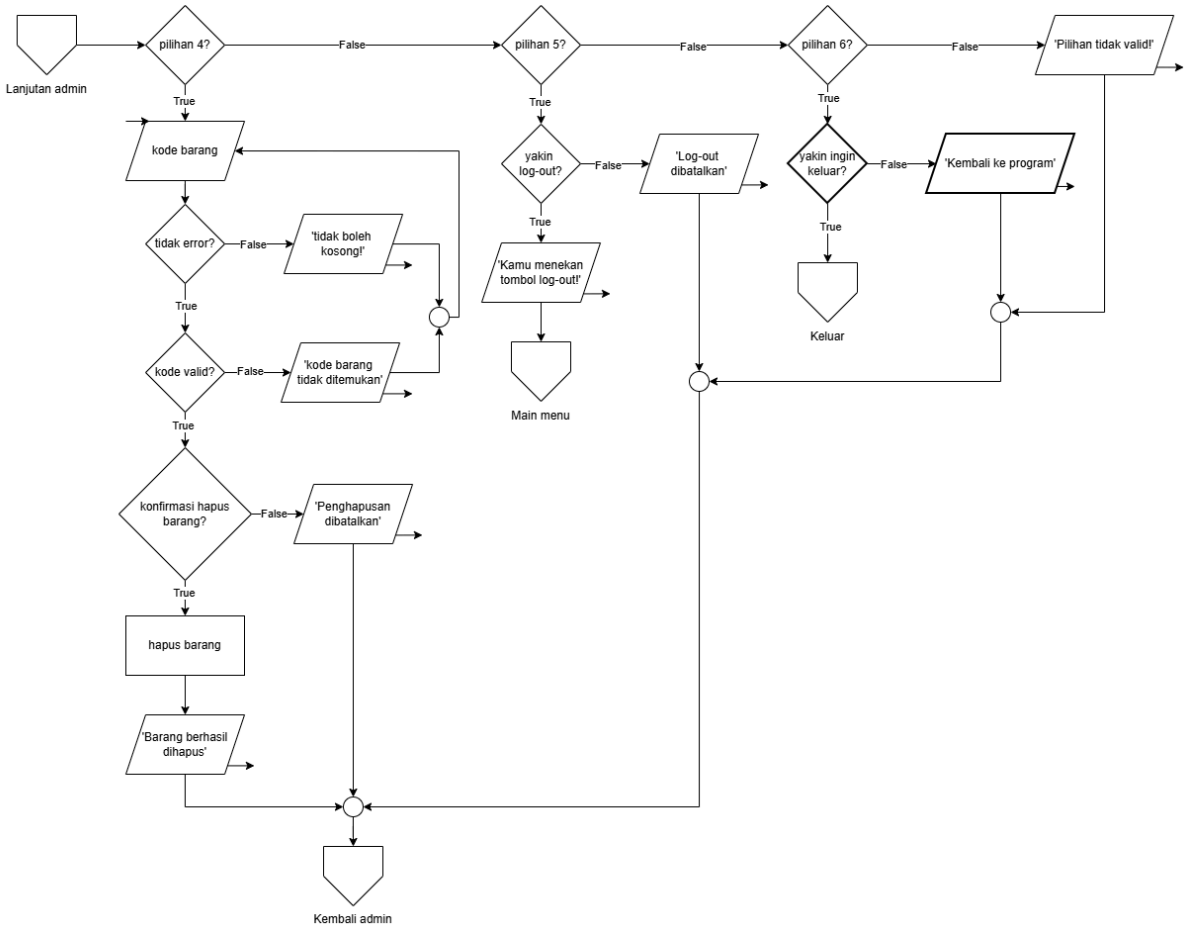
Memilih 3 atau keluar maka program akan meminta konfirmasi sebelum betul-betul keluar dari program. *Input* yang tidak valid akan membawa kembali pengguna ke *main menu*.



Gambar 1.2 Flowchart Menu admin

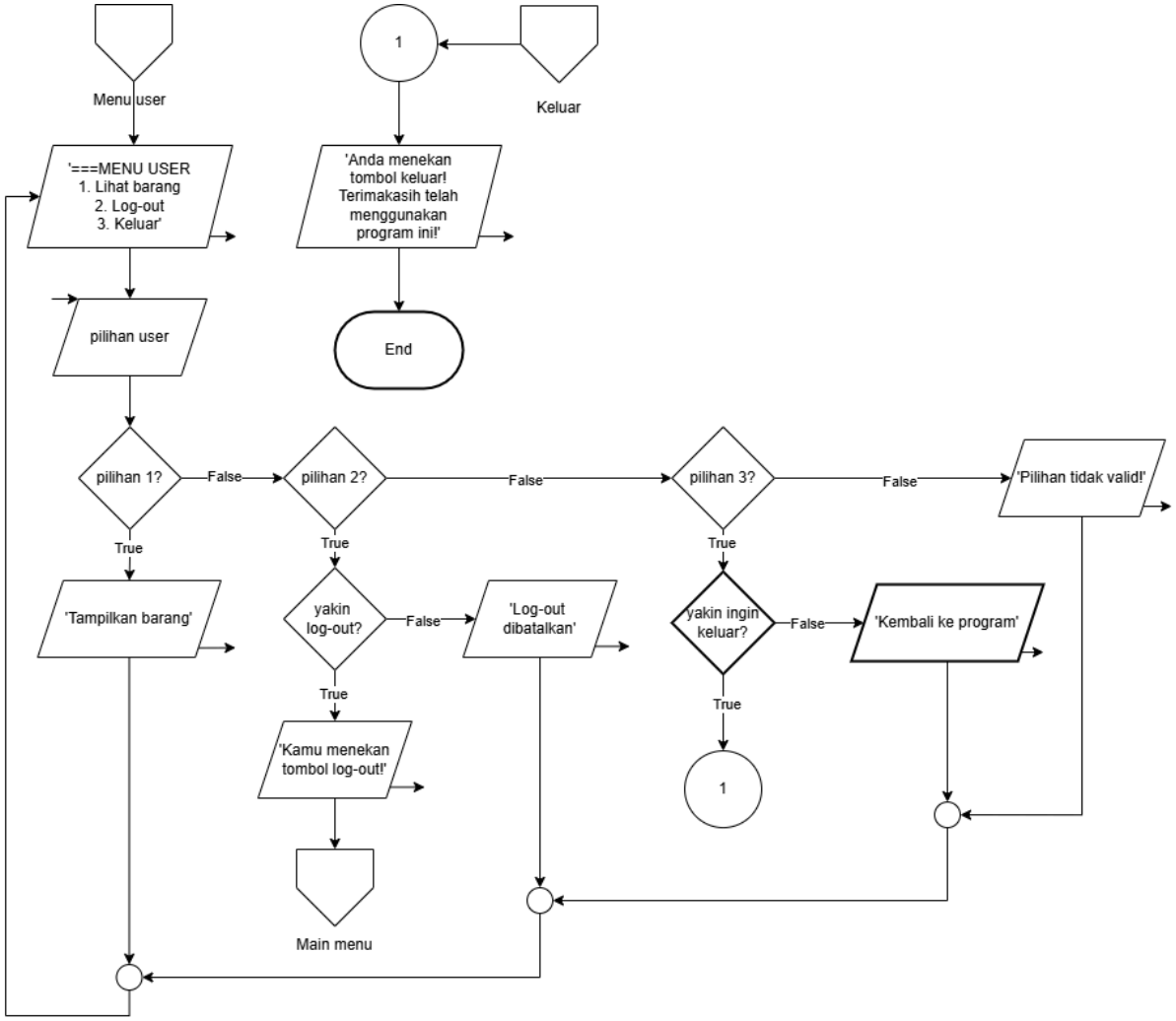
Pada *Gambar 1.2*, setelah berhasil login, *role* dari *username* akan dibaca. Seorang admin dapat melakukan 4 hal, melihat barang, menambah barang, mengubah data barang,

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. Pengertian Fungsi dan Prosedur

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 3.1.1 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. Adapun prosedur yaitu blok kode seperti fungsi namun tidak mengembalikan nilai, walaupun ada *return* di akhir blok.

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* atau alias.

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

Gambar 3.1.2 Memanggil Fungsi yang Berbeda File

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

Gambar 3.1.3 Memanggil Fungsi yang Berbeda File dengan Nama Alias

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.

B. Dictionary - kamus.py

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.

```
#Variabel global
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 3.2.1 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’.

C. Daftar Fungsi dan Prosedur

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

Gambar 3.3.1 Impor Library dan Impor File

Sebelum memulai fungsi, *library os* dan *time* akan diimpor terlebih dahulu. Lalu *library* eksternal *prettytable* serta ‘inventaris’ dan ‘users’ dari *file* kamus.py

a. Prosedur Membersihkan Terminal - clear()

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

Gambar 3.3.1.1 Prosedur Membersihkan Terminal

Seperti sebelumnya, prosedur ini sesederhana membersihkan terminal, bisa digunakan untuk sistem operasi seperti Windows, macOS, dan Linux.

b. Prosedur Menjeda Waktu - delay()

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

Gambar 3.3.2.1 Prosedur Memberi Jeda Waktu

Prosedur ini akan memberi jeda waktu sebelum mengeksekusi kode berikutnya.

c. Fungsi Menampilkan Tabel Barang - daftar()

```
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 3.3.3.1 Fungsi Menampilkan 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. Ini merupakan

contoh fungsi **tanpa parameter**. Parameter adalah variabel dalam definisi fungsi yang bertindak sebagai *placeholder* untuk nilai yang akan diterima, sedangkan argumen adalah nilai aktual yang diteruskan ke fungsi saat dipanggil.

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 ini dipanggil.

d. Fungsi Rekursif - tampil_menu_rekursif()

```
def tampil_menu_rekursif(menu_list, index = 0):
    #Base Case
    if index >= len(menu_list):
        return
    #Recursive Case
    print(f'{index + 1}. {menu_list[index]}')
    tampil_menu_rekursif(menu_list, index + 1)
```

Gambar 3.3.4.1 Fungsi Rekursif dan dengan Parameter.

Fungsi ini merupakan jenis fungsi rekursif, yaitu fungsi yang memanggil dirinya sendiri. Dalam rekursif, terdapat dua bagian, pertama adalah *base case*, yaitu kondisi di mana fungsi berhenti memanggil dirinya sendiri. Lalu ada *recursive case*, ini adalah bagian di mana fungsi memanggil dirinya sendiri lagi, tetapi dengan input yang telah dimodifikasi agar semakin mendekati *base case*. Dalam fungsi ini, ada parameter ‘menu_list’ dan ‘index’ yang bernilai 0.

Dalam *base case* di atas, fungsi akan berhenti memanggil dirinya sendiri jika jumlah ‘index’ lebih besar atau sama dengan jumlah indeks pada *list* yang akan digunakan untuk fungsi ini.

Di *recursive case*, ada sintaks *print* yang akan menampilkan angka dan isinya, ini hanya sekadar menampilkan *menu* yang diawali dengan angka supaya pengguna tahu *menu* ini ada di angka/urutan ke berapa. Lalu fungsi rekursif berjalan kembali dengan menambah 1 pada ‘index’ hingga *base case* terpenuhi. Berikut contoh penerapan fungsi ini:

```
print('===MENU ===')
menu = ['Lihat barang', 'Tambah barang', 'Log-out']
tampil_menu_rekursif(menu)
```

Gambar 3.3.4.2 Contoh Penerapan Fungsi Rekursif

Variabel ‘menu’ berisi tiga indeks, kemudian fungsi rekursif akan dipanggil dengan variabel tersebut akan menjadi argumen untuk parameter ‘menu_list’ pada fungsi. Cara jalannya akan dimulai dari indeks 0 atau “Lihat barang” dan seterusnya, jika jumlah ‘index’ sama dengan jumlah indeks pada ‘menu’, maka *base case* akan dijalankan dan fungsi rekursif berakhir.

e. Fungsi Error Handling Mencegah Input Kosong - input_str(pesan)

```
def input_str(pesan): #Fungsi erorr handling untuk mencegah input kosong
    while True:
        try:
            fakta = input(pesan)
            if fakta.strip() == '':
                raise ValueError('Input tidak boleh kosong!')
            return fakta
        except ValueError as e:
            print(e)
```

Gambar 3.3.5.1 Error Handling untuk Input Kosong String

Fungsi *error handling* ini bertujuan untuk mencegah *input* kosong dengan menggunakan mekanisme *try-except*.

Blok *try* digunakan untuk mengelompokkan potongan kode yang mungkin menyebabkan *exception*, sementara blok *except* memberikan penanganan khusus untuk setiap jenis *exception* yang mungkin muncul.

Dalam fungsi ini, blok *try* akan meminta *input* dari pengguna dan menyimpan hasilnya ke dalam variabel 'fakta'. Kemudian dilakukan pengecekan apakah 'fakta' kosong atau tidak setelah dihilangkan spasi menggunakan method strip().

Jika input kosong, program akan melemparkan *error* (*raise*) bertipe *ValueError* dengan pesan 'Input tidak boleh kosong'. *Error* yang dilempar ini kemudian ditangkap oleh blok *except* dan disimpan dalam variabel 'e', lalu pesan *error* tersebut ditampilkan dengan *print*(e).

Jika *input* tidak kosong, maka *raise ValueError* tidak dijalankan dan fungsi akan mengembalikan (*return*) nilai 'fakta' yang berisi *input* dari pengguna.

f. Fungsi Error Handling Untuk Input Password Akun Baru- regist_pw(pesan)

```
def regist_pw(pesan):
    while True:
        try:
            pw_baru = input(pesan)
            if len(pw_baru) < 3:
                raise ValueError('Password minimal 3 karakter!')
            return pw_baru
        except ValueError as e:
            print(e)
```

Gambar 3.3.6.1 Error Handling untuk Register

Fungsi ini akan digunakan saat mengeksekusi *menu register* di main.py. Fungsi ini menggunakan perulangan *while*, jadi blok kode akan terus berjalan hingga *input* sesuai.

Error handling ini digunakan untuk mencegah pengguna *menginput password* yang kurang dari tiga karakter. Ini juga multifungsi jika pengguna *menginput* nilai kosong, program akan melempar *raise* dengan tipe *ValueError*.

g. Fungsi Error Handling Untuk Jumlah dari Barang Baru - jumlah(pesan)

```
def jumlah(pesan):
    while True:
        try:
            jumlah_input = input(pesan)
            if jumlah_input.strip() == '':
                raise ValueError('Input tidak boleh kosong!')
            jumlah_int = int(jumlah_input)
            if jumlah_int <= 0:
                raise ValueError('Jumlah harus lebih besar dari 0!')
            return jumlah_int
        except ValueError:
            if jumlah_input.strip() == '':
                print('Input tidak boleh kosong!')
            else:
                try:
                    if int(jumlah_input) <= 0:
                        print('Jumlah harus lebih besar dari 0!')
                except:
                    print('Input harus berupa angka!')
```

Gambar 3.3.7.1 Error Handling Input Jumlah Barang

Fungsi *error handling* ini akan digunakan saat pengguna memilih *menu* menambah barang baru tepatnya saat diminta *menginput* jumlah barang tersebut.

Disini terdapat dua *raise ValueError*, kondisi pertama yaitu *input* tidak boleh kosong. Jika ini terpenuhi, variabel ‘jumlah_input’ yang awalnya berupa *string* angka akan berubah menjadi *integer.*

ValueError berikutnya untuk mencegah jumlah *diinput* “0”. Jika salah satu *error* terjadi, *raise ValueError* yang akan ditangkap oleh *except* dan pesan yang sesuai ditampilkan.

h. Fungsi Login - login()

```
def login(): #Fungsi untuk login
    clear()
    print('Login akun')
    while True:
        #Variabel lokal
        usr = input_str('Masukkan Username\t:')
        pw = input_str('Masukkan password\t:')

        for u in users:
            if usr == u and pw == users[u]['password']:
                role = users[u]['role']
                clear()
                print('Login berhasil!')
                delay()
                return usr, role
            else:
                clear()
                print('Username atau passsword salah!')
```

Gambar 3.3.8.1 Fungsi Login

Dalam fungsi ini, pengguna diminta *menginput* dua variabel lokal ‘usr’ dan ‘pw’, logika *input* ini memanggil fungsi ‘input_str’ 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*. Terakhir nilai dari kedua variabel lokal akan dikembalikan (*return*).

i. Fungsi Registrasi - register()

```
def register(): #Fungsi untuk registrasi
    clear()
    print('Register')
    while True:
        cek_ada_usr = False
        usr_baru = input_str('Masukkan username baru:')

        for u in users:
            if usr_baru == u:
                cek_ada_usr = True
                break
        if cek_ada_usr:
            print('Username sudah terpakai! Gunakan username lain')
        else:
            pw_baru = regist_pw('Masukkan password:')
            users.update({
                        usr_baru : {'password' : pw_baru,
                                    'role' : 'user'
                        }
                    })
            clear()
            print('Registrasi berhasil!')
            delay()
            return usr_baru, pw_baru
```

Gambar 3.3.9.1 Fungsi Registrasi

Proses awal dari fungsi ini yaitu pencegahan *username* baru sama dengan *username* yang sudah ada. Seperti sebelumnya ‘u’ berisi *key* dari *dict* ‘users’, jika ada yang sama maka nilai ‘cek_ada_usr’ menjadi *true* dan perulangan di-*break*.

Program akan kembali ke proses *input username* jika *username* baru telah digunakan. Jika tidak lanjut ke *input password* dengan memanggil fungsi ‘regist_pw()’. 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. Ini juga merupakan contoh fungsi tanpa parameter.

j. Fungsi Keluar Program - keluar()

```
def keluar(): #Fungsi untuk keluar dari program
    clear()
    while True:
        konfirmasi = input('Yakin ingin keluar dari program?(y/n):')

        if konfirmasi.lower() == 'y':
            clear()
            print('Anda menekan tombol keluar! Terimakasih telah menggunakan
program!')
            delay()
            exit()
        elif konfirmasi.lower() == 'n':
            print('Kembali ke program')
            delay()
            clear()
            return
        else:
            print('Input tidak valid!')
```

Gambar 3.3.10.1 Fungsi Keluar dari Program

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*.

k. Prosedur Tambah Barang - tambah_barang()

```
#Contoh prosedur
def tambah_barang(): #Prosedur untuk menambah barang baru ke daftar barang
    while True:
        nama_barang = input_str('Nama barang:')

        while True:
            cek_kode = False
            kode_barang = input_str('Kode barang:')

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

```
        jumlah_barang = jumlah('Jumlah barang:')

        kategori_barang = input_str('Kategori barang\t:')

        kondisi_barang = input_str('Kondisi barang\t:')

        inventaris.update({
            kode_barang : {'nama' : nama_barang,
                           'jumlah' : jumlah_barang,
                           'kategori' : kategori_barang,
                           'kondisi' : kondisi_barang}
        })

        clear()
        print('Barang berhasil ditambahkan!')
        delay()
        return
```

Gambar 3.3.11.1 Prosedur Menambah Barang Baru

Berikutnya ada prosedur jika admin sudah *login*, 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.

Untuk jumlah barangnya, kita hanya perlu memanggil prosedur ‘jumlah()’, dan sisanya dengan prosedur ‘input_str()’. Setelah itu *dict* akan *diupdate*. *Return* disini hanya mengembalikan pengguna kembali ke *menu* admin nantinya setelah proses penambahan barang baru selesai.

l. Prosedur Ubah Data Barang - ubah_data()

```
def ubah_data(): #Fungsi mengubah data barang yang sudah ada
    ditemukan = False
    clear()
    print('Ubah data barang')
    print(daftar())

    cari_kode = input_str('Masukkan kode barang yang ingin diubah\t:')
    clear()

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

Gambar 3.3.12.1 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('Pilih data yang ingin diubah:')
            menu_ubah_barang = ['Ubah nama barang', 'Ubah jumlah barang',
                                'Ubah kategori barang', 'Ubah kondisi
barang']
            tampil_menu_rekursif(menu_ubah_barang)

            pilihan_ubah = input('Pilih apa yang ingin diubah: ')
```

Gambar 3.3.12.2 Contoh Penerapan Fungi Rekursif

Disini kita bisa menggunakan fungsi rekursif sebelumnya untuk menampilkan *menu* apa saja yang bisa dilakukan, seperti yang dijelaskan sebelumnya, *list* ini memiliki empat indeks, setiap indeks akan ditampilkan hingga ke indeks 3 atau ‘Ubah kondisi barang’.

```
if pilihan_ubah == '1':
                nama_baru = input_str('Ubah nama barang:')
                inventaris[k]['nama'] = nama_baru
                clear()
                print('Nama berhasil diubah!')
                delay()
                return
```

Gambar 3.3.12.3 Mengubah 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’ akan berubah setelah *menginput* nama baru. Sama seperti sebelumnya, *return* di akhir akan membawa pengguna kembali ke *menu* admin.

```
if pilihan_ubah == '2':
                clear()

                print(f"Jumlah barang saat ini : {ada['jumlah']}")
                print('Pilih:')
                pilihan = ['Tambah jumlah barang', 'Kurangi jumlah barang']
                tampil_menu_rekursif(pilihan)

                while True:
                            tamkur = input('Pilih (1/2): ')
                            if tamkur in ['1', '2']:
                                break
                            else:
                                print('Pilihan tidak valid!')

                while True:
                    try:
                        jumlah_tamkur = int(input('Masukkan jumlah: '))
                        if jumlah_tamkur <=0:
                            print('Jumlah harus lebih dari 0!')
                            continue

                        if tamkur == '1':
                            ada['jumlah'] += int(jumlah_tamkur)
                            clear()
                            print('Jumlah barang berhasil ditambahkan!')

                        else:
                            if ada['jumlah'] >= int(jumlah_tamkur):
                                ada['jumlah'] -= int(jumlah_tamkur)
                                clear()
                                print('Jumlah barang berhasil dikurangi!')
                            else:
```

```
                                print('Jumlah terlalu besar untuk
dikurangi')
                                continue
                        delay()
                        return
                    except ValueError:
                        print('Harus berupa angka lebih dari 0!')
```

Gambar 3.3.12.4 Mengubah 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.

Setelah itu, kita menggunakan *try-except* lagi. Di blok *try*, *if* pertama akan dieksekusi jika ‘jumlah_tamkur’ kurang dari atau sama dengan “0”.

Jika tidak, lanjut ke blok berikutnya yaitu proses penjumlahan dan pengurangan jumlah barang, khusus pengurangan, jika pengguna *menginput* ‘jumlah_tamkur’ melebihi jumlah barang saat ini, pesan “Jumlah terlalu besar untuk dikurangi” akan ditampilkan dan kembali *menginput* nilai yang sesuai, lalu kembali ke *menu* admin.

Di blok *except*, jika *input* bukan berupa angka, pesan yang ada di blok tersebut akan ditampilkan.

```
elif pilihan_ubah == '3':
                kategori_baru  = input_str('Ubah kategori barang\t:')
                inventaris[k]['kategori'] = kategori_baru
                clear()
                print('Kategori barang berhasil diubah')
                delay()
                return

            elif pilihan_ubah == '4':
                kondisi_baru  = input_str('Ubah kondisi barang\t:')
                inventaris[k]['kondisi'] = kondisi_baru
                print('Kondisi barang berhasil diubah')
                delay()
                return

            else:
                print('Input tidak valid!')
                delay()
                return

    if not ditemukan:
        print('Kode barang tidak ditemukan')
```

```
        delay()
        return
```

Gambar 3.3.12.5 Mengubah Kategori dan Kondisi Barang, Pilihan Tidak Valid, dan Kode Tidak Ditemukan

Blok berikutnya hanya pilihan mengubah kategori dan kondisi barang seperti saat mengubah nama barang. Lalu jika pilihan tidak valid akan kembali ke *menu* admin, dan jika kode barang tidak ditemukan, juga akan kembali ke *menu* admin.

m. Prosedur Hapus Barang - hapus_barang()

```
def hapus_barang(): #Fungsi menghapus barang
    while True:
        ketemu = False
        clear()
        print('Hapus barang')
        print(daftar())

        cari_kode = input_str('Masukkan kode barang yang ingin diubah\t:')
        clear()


        for l in inventaris:
            if l == cari_kode:
                ketemu = True
                ada = inventaris[l]
                print(f'''Data ditemukan:
Nama\t\t: {ada['nama']}
Kode\t\t: {l}
Jumlah\t\t: {ada['jumlah']}
Kategori\t: {ada['kategori']}
Kondisi\t\t: {ada['kondisi']}''')

                hapus = input('Yakin ingin menghapus barang?(y/n)')

                if hapus.lower() == 'y':
                    inventaris.pop(l)
                    clear()
                    print('Barang berhasil dihapus!')
                    delay()
                    return
                elif hapus.lower() == 'n':
                    print('Penghapusan dibatalkan')
                    delay()
                    return
                else:
```

```
                    print('Input tidak valid!')
                    delay()
                    return

        if not ketemu:
            print('Kode barang tidak ditemukan!')
        lagi = input('Ingin hapus barang lagi?(y/n):')
        if lagi.lower() != 'y':
            clear()
            delay()
            return
        clear()
```

Gambar 3.3.13.1 Prosedur Menghapus Barang

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. Kondisi berikutnya jika kode barang tidak ditemukan, akan ada opsi untuk meminta pengguna mengkonfirmasi apakah ingin menghapus barang lagi, jika tidak maka kembali ke *menu* admin dan kembali ke fungsi jika *menginput* 'y'.

n. Fungsi Log-out - logout()

```
def logout(): #Fungsi log-out dari menu admin/user
    clear()
    kembali = input('Yakin ingin log-out?(y/n):')
    if kembali.lower() == 'y':
        clear()
        print('Kamu menekan tombol log-out!')
        delay()
        return True
    elif kembali.lower() == 'n':
        print('Log-out dibatalkan')
        delay()
        return False
    else:
        print('Pilihan tidak valid!')
        delay()
        return False
```

Gambar 3.3.14.1 Fungsi Kembali ke Main Menu

Dalam fungsi ini, sebelum kembali ke *main menu*, program akan meminta konfirmasi. Jika pengguna *menginput* selain 'y', nilai *return* menjadi *True.*

D. File Utama

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

Gambar 3.4.1 Impor File

Sebelum mengerjakan *file* utama, semua *file-file* yang dipisah tadi harus diimpor terlebih dahulu, fungsimajor.py akan dialiaskan sebagai ‘sip’.

```
while True: #Loop utama program
    #Variabel global
    role = None
    loginn = False
    while True:
        print('===Menu===')
        menu_utama = ['Login', 'Registrasi', 'Keluar']
        sip.tampil_menu_rekursif(menu_utama)

        menu = input('Pilih menu:')

        if menu == '1':
            usr, role = sip.login()
            loginn = True
            break
```

Gambar 3.4.2 Blok Kode Login

Sebelum memulai, ada dua *flag* yaitu ‘role’ bernilai *none* dan ‘loginn’ bernilai *False.* Fungsi rekursif digunakan kembali untuk menampilkan pilihan *main menu*. Lalu permintaan *input*, jika memilih ‘1’ atau *login*, fungsi login dari ‘sip’ akan dipanggil. Setelah selesai login, ‘loginn’ menjadi *true* dan lanjut ke *menu* yang sesuai dengan *role*.

```
elif menu == '2':
            print('Registrasi')
            usr_baru, pw_baru  = sip.register()
            sip.clear()
            continue
        elif menu == '3':
            sip.keluar()
        else:
            print('Input tidak valid!')
    log_out = False
```

Gambar 3.4.3 Blok Kode Register

Kode ini memanggil fungsi ‘sip.register()’, setelah selesai registrasi, pengguna akan kembali ke *main menu* karena menggunakan *continue*. Lalu ada pilihan keluar dan input tidak valid.

```
while True:
        if role == 'admin': #Menu admin
            print(f'Halo, {usr}!')

            print('===MENU ADMIN===')
            menu_utama_admin = ['Lihat barang', 'Tambah barang', 'Ubah data
barang',
                                'Hapus barang', 'Log-out', 'Keluar']
            sip.tampil_menu_rekursif(menu_utama_admin)

            menu_admin = input('Pilih menu:')

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

            elif menu_admin == '2':
                sip.clear()
                print('Tambah barang')
                sip.tambah_barang()

            elif menu_admin == '3':
                sip.clear()
                print('Ubah data barang')
                sip.ubah_data()

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

```
                sip.hapus_barang()
```

Gambar 3.4.4 Menu Admin

Gambar di atas merupakan empat pilihan yang bisa dilakukan admin, ada lihat barang, menambah barang, mengubah data barang, dan menghapus barang yang di mana sudah dijelaskan di sub-bab sebelumnya. Yang dilakukan hanya memanggil fungsi/prosedur tersebut sesuai dengan alias yang sudah ditetapkan.

```
elif menu_admin == '5':
                if sip.logout():
                    log_out = True
                    break
```

Gambar 3.4.5 Opsi Log-out

Disini fungsi *log-out* digunakan, sebelumnya jika betul-betul kembali ke *main menu*, fungsi ini berarti berhasil dijalankan. Nilai *true* dikembalikan dan ‘if sip.logout()’ akan dijalankan, nilai ‘log_out’ yang awalnya *false* berubah menjadi *true* dan kembali kembali ke *main menu* karena *break*.

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

            else:
                sip.clear()
                print('Input tidak valid!')
            continue
        if log_out:
            break
```

Gambar 3.64.6 Opsi Lainnya

Pilihan keenam memanggil fungsi keluar untuk keluar dari program, *error handling* jika input di luar pilihan, lalu *continue* untuk mengembalikan pengguna ke *menu* admin setelah selesai memproses salah satu opsi.

Sebelumnya jika ‘log_out’ bernilai *true*, *menu* admin akan diberhentikan dan kembali ke *main menu*.

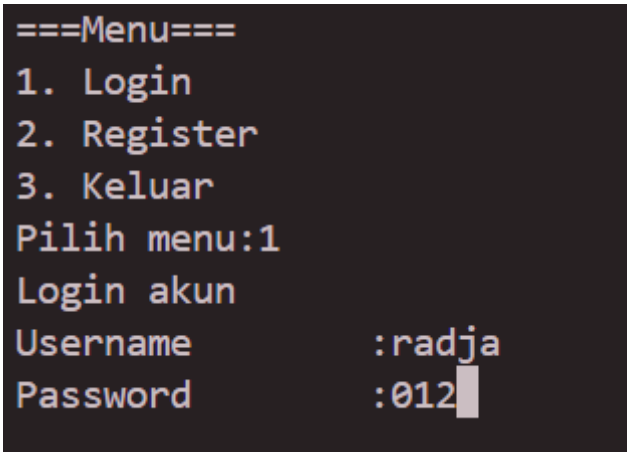
```
else:
            print(f'Halo, {usr}!')
            print('===MENU USER===')

            menu_utama_user = ['Lihat barang', 'Log-out', 'Keluar']
            sip.tampil_menu_rekursif(menu_utama_user)

            menu_user = input('Pilih menu: ')

            if menu_user == '1':
                sip.clear()
                print('Lihat barang')
                print(sip.daftar())
            elif menu_user == '2':
                if sip.logout():
                    log_out = True
                    break
            elif menu_user == '3':
                sip.clear()
                sip.keluar()
            else:
                print('Input tidak valid!')
        if log_out:
            break
```

Gambar 3.4.7 Menu User

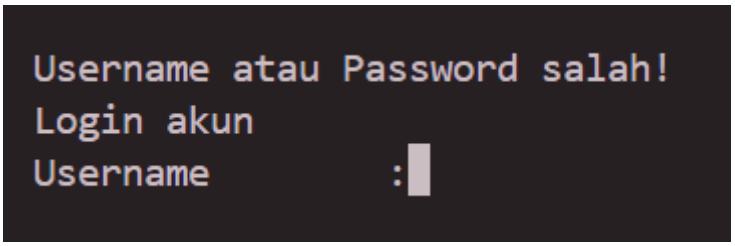
Jika pengguna *login* sebagai *user*, mereka hanya memiliki tiga pilihan, yaitu lihat barang, *log-out*, dan keluar dari program, dengan menggunakan kode yang sama, *menu user* secara keseluruhan sangat sederhana dan tidak rumit untuk dipahami.

Fungsi *log-out* juga dijalankan dengan cara yang sama seperti di *menu* admin agar pengguna kembali ke *main menu*.

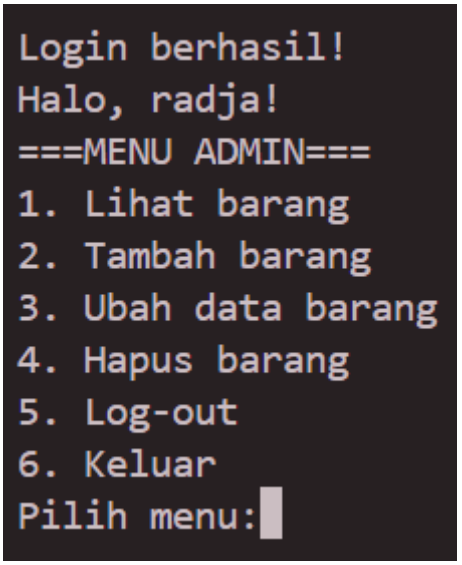
4. Hasil Output



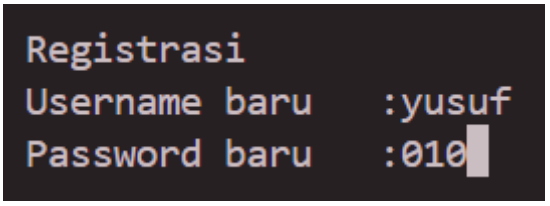
Gambar 4.1 Login



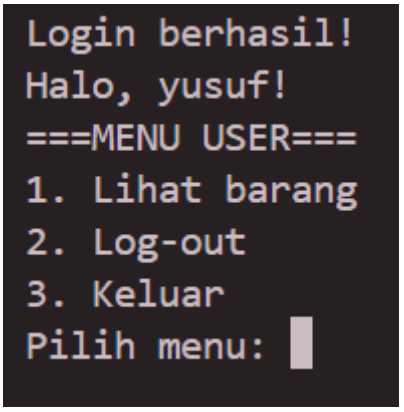
Gambar 4.2 Username atau Password Salah



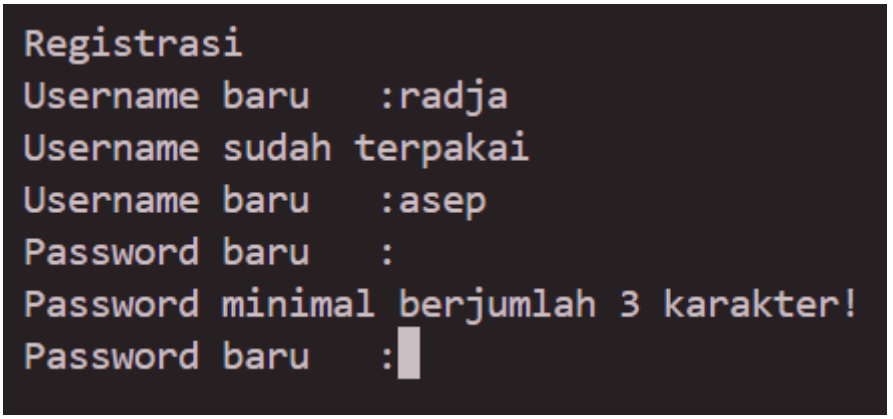
Gambar 4.3 Login Sebagai radja dan Berhasil



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
```

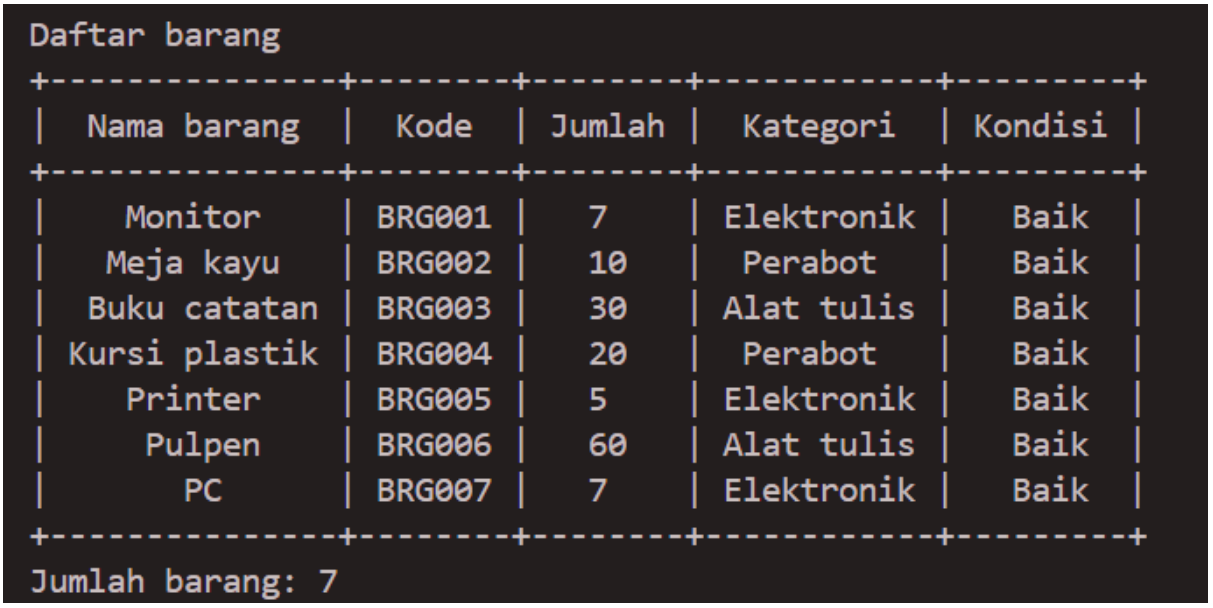
Gambar 4.7 Proses Menambahkan Barang

```
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)

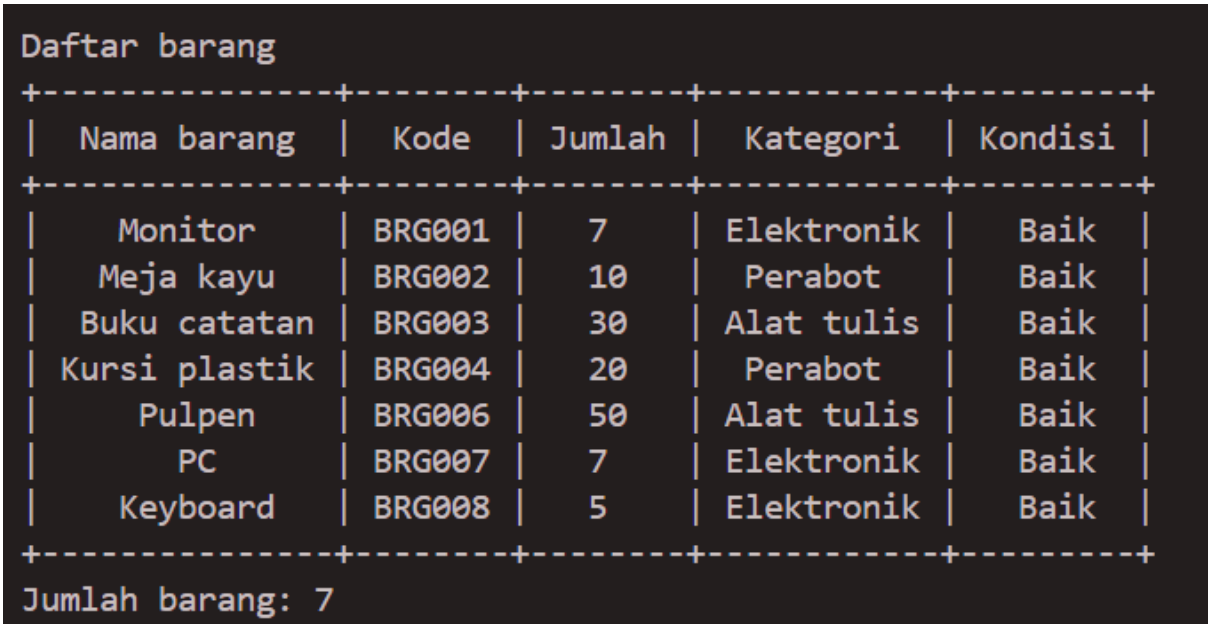
```
Daftar barang
+---------------+--------+--------+------------+---------+
|  Nama barang  |  Kode  | Jumlah |  Kategori  | Kondisi |
+---------------+--------+--------+------------+---------+
|    Monitor    | BRG001 |   7    | Elektronik |   Baik  |
|   Meja kayu   | BRG002 |   10   |  Perabot   |   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



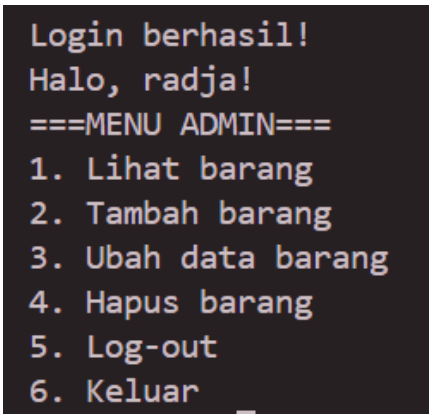
```
Daftar barang
+--------------+--------+--------+------------+---------+
|  Nama barang |  Kode  | Jumlah |  Kategori  | Kondisi |
+--------------+--------+--------+------------+---------+
|    Monitor   | BRG001 |   7    | Elektronik |   Baik  |
|   Meja kayu  | BRG002 |   10   |  Perabot   |   Baik  |
|  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



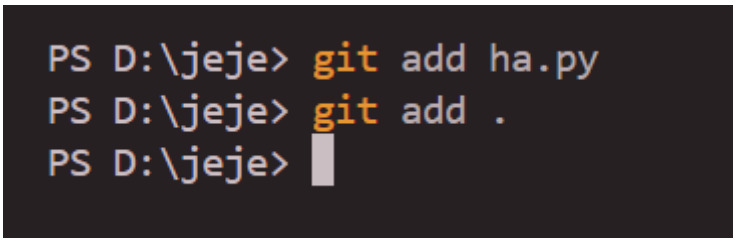
```
Login berhasil!
Halo, radja!
===MENU ADMIN===
1. Lihat barang
2. Tambah barang
3. Ubah data barang
4. Hapus barang
5. Log-out
6. Keluar
```

Gambar 4.12 Output Fungsi Rekursif

5. Langkah-Langkah GIT

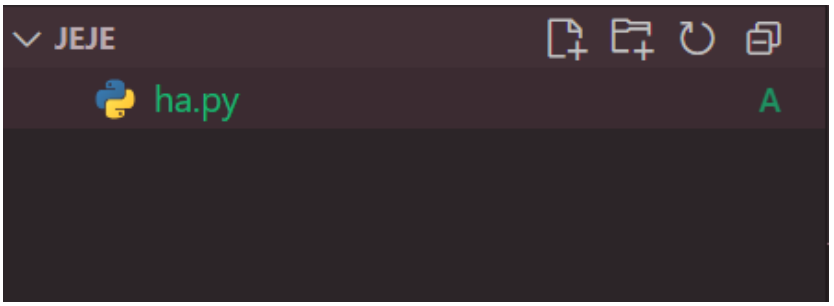
Jika dalam suatu folder kita sudah pernah melakukan *git init* dan *git remote* maka kita tidak perlu lagi melakukannya dan langsung ke *git add*.

5.1 GIT Add



Gambar 5.1 Proses Git Add

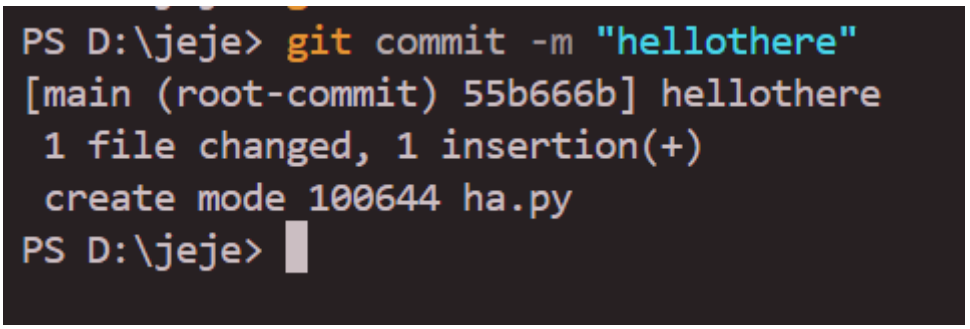
Git add berfungsi untuk memindah semua perubahan di area kerja ke indeks, ada dua cara untuk melakukan *git add*, yang pertama dengan mengetikkan nama *file* yang ingin di *add*. Jika kalian ingin menambahkan banyak *file* sekaligus, ketik "git add ."



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

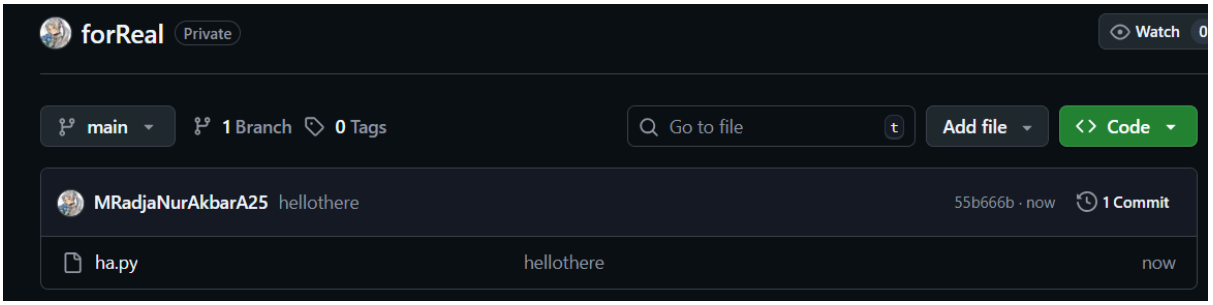
Git commit berfungsi untuk mengkonfirmasi setiap perubahan pada *repository* kalian dengan mengetikkan “git commit -m ‘pesan yang ingin ditulis’ ”

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