




# **PROGRAM BUDGETING DAN PENCATATAN KEUANGAN PRIBADI**

DITUJUKAN UNTUK MEMENUHI  
PROJECT TEAM PROGRAM  
KOMPUTER KELOMPOK 6

Disusun oleh :

Anggito Dwi Prihantoro (I0322015)  
Ariiq Rahmat Setiana (I0322017)  
Bagus Rakha Pramuditya (I0322023)  
Deanieta Adilest (I0322034)



## **BAB I**

### **DESKRIPSI MASALAH**

Pengelolaan keuangan pribadi merupakan sebuah persoalan yang mencakup beberapa aspek yang perlu diatasi. Kesulitan dalam pencatatan uang dengan multi-dompet seperti tunai, e-money, dan rekening bank seringkali membuat pengguna kesulitan dalam mengintegrasikan dan menyatukan catatan keuangan dari berbagai sumber ini, sehingga sulit melacak dan mencatat semua transaksi dengan tepat waktu. Selain itu, keterlambatan dalam mencatat pemasukan dan pengeluaran uang juga menjadi persoalan umum.

Saat pengguna memiliki kegiatan yang padat, seringkali pengguna merasa kesulitan untuk menghentikan aktivitas dan mencatat setiap transaksi secara detail, sehingga informasi keuangan tidak tercatat dengan cepat dan tepat. Masalah lainnya yang sering ditemui pengguna adalah kurangnya kemampuan untuk mencatat utang secara langsung dan segera menampilkan jumlah utang yang harus dibayar. Selain itu ada masa di mana pengguna menggunakan pengeluaran yang terlalu banyak dalam satu waktu. Pengguna membutuhkan sistem yang dapat memberikan peringatan ketika salah satu tipe pengeluaran melebihi batas yang telah ditentukan sebelumnya.

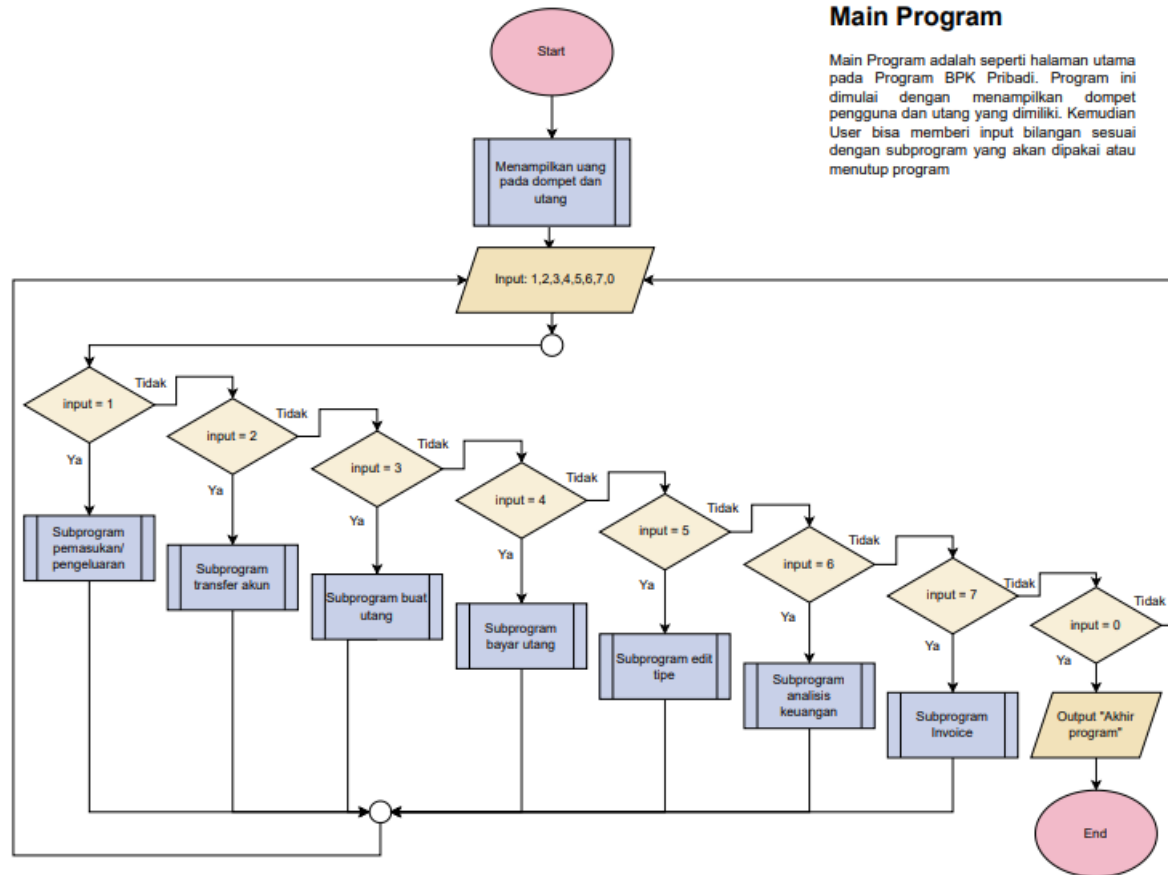
Menurut permasalahan di atas, pengguna membutuhkan solusi yang dapat membantu mereka dalam mencatat utang secara cepat dan memberikan informasi yang jelas mengenai jumlah utang. Selain itu, peringatan pengeluaran yang melebihi batas juga menjadi masalah yang perlu diatasi. Dengan adanya peringatan tersebut, pengguna dapat lebih mudah mengontrol pengeluaran dan menjaga agar keuangan tetap seimbang. Pengguna membutuhkan solusi yang memudahkan mereka untuk mencatat dan melacak semua aktivitas keuangan dalam satu bulan, sehingga mereka dapat memiliki informasi yang lebih jelas dan mudah dalam merencanakan keuangan mereka.

Dalam rangka mengatasi masalah-masalah tersebut, diperlukan solusi yang efektif seperti program pencatatan transaksi dari berbagai dompet, pencatatan pemasukan dan pengeluaran uang dengan cepat, serta pemantauan pengeluaran yang melebihi batas yang ditentukan. Pengguna juga membutuhkan program yang memudahkan pencatatan utang secara langsung dan memberikan informasi utang yang jelas. Kemudian, program yang memfasilitasi pelacakan dan pencatatan keuangan dalam satu bulan dengan invoice akan sangat berguna untuk pengelolaan keuangan yang efektif dan teratur.

## BAB II

### FLOWCHART

#### 2.1 Main Program



Main program menampilkan dompet dan utang (melalui subprogram) dan meminta input pengguna untuk memilih subprogram yang ada, alur dari main program adalah sebagai berikut:

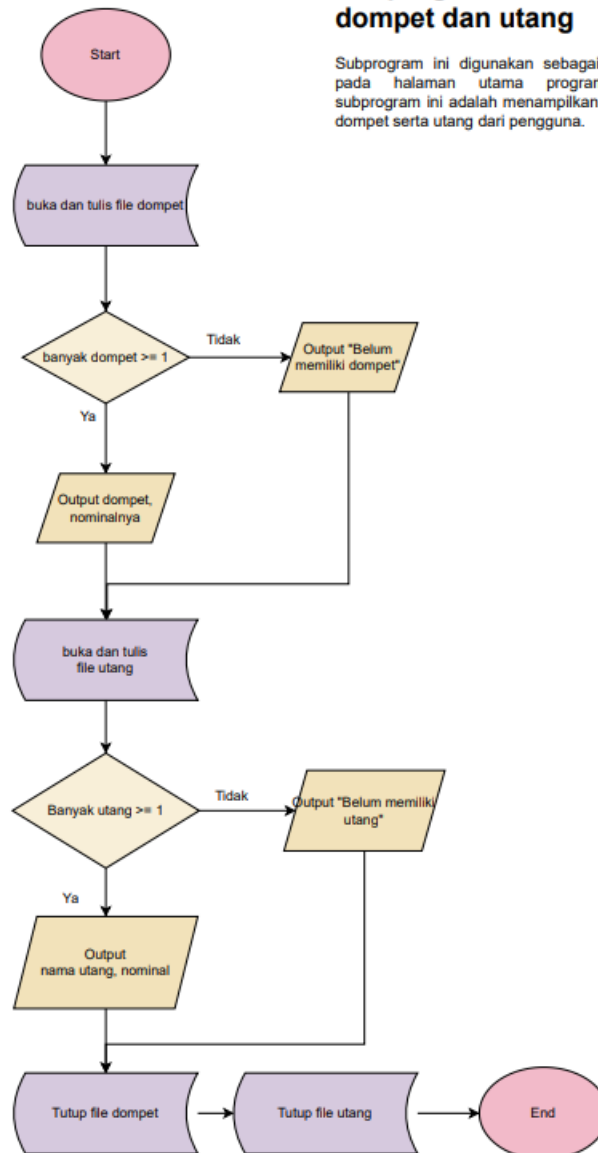
1. Saat program dijalankan, menjalankan subprogram menampilkan dompet dan utang untuk menampilkan dompet dan utang yang dimiliki pengguna.
2. Meminta input bilangan bulat pada user untuk memilih subprogram, diantaranya: 1. Subprogram pemasukan/pengeluaran, 2. Subprogram transfer akun, (terusin punyaku), 7. Subprogram invoice, 0. Keluar dari program. Ini adalah halaman utama dari program BPK Pribadi.
3. Ketika pengguna memasukkan input 1 sampai 6, program akan menjalankan subprogram pilihan dan ketika subprogram selesai, mengembalikan pengguna ke halaman utama.
4. Ketika input 0 diberikan, main program diakhiri dan program berhenti.

## 2.2 Menampilkan Dompot dan Utang

Subprogram Menampilkan dompet dan utang

### Subprogram Menampilkan dompet dan utang

Subprogram ini digunakan sebagai pengantar pada halaman utama program. Fungsi subprogram ini adalah menampilkan uang pada dompet serta utang dari pengguna.



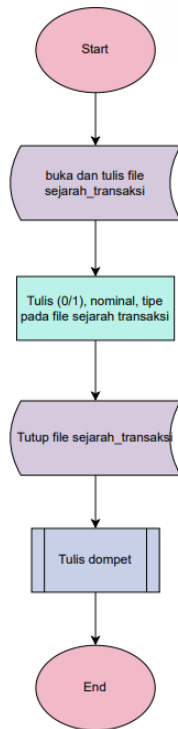
Subprogram menampilkan dompet dan utang berfungsi untuk menampilkan uang pada dompet serta utang dari pengguna. Subprogram akan menampilkan utang atau dompet jika ada. Alur dari subprogram menampilkan dompet dan utang adalah sebagai berikut:

1. Program dimulai
2. Pengguna membuka dan menulis file dompet
3. Apabila dompet lebih dari atau sama dengan 1 maka akan muncul dompet dan nominalnya, apabila dompet sama dengan 0 maka akan muncul output belum memiliki dompet.

4. Kemudian pengguna akan membuka dan menulis file utang
5. Apabila banyak utang lebih dari atau sama dengan 1 maka akan muncul nama utang dan nominalnya, jika banyak utang sama dengan 0 maka akan muncul output belum memiliki utang.
6. Terakhir tutup file dompet dan utang.

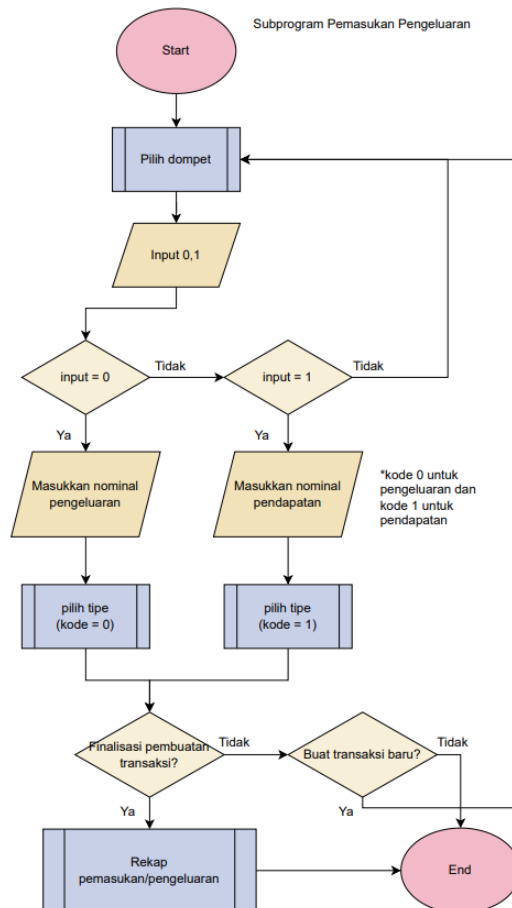
### 2.3 Pemasukan dan Pengeluaran

Subprogram Rekap Pemasukan/Pengeluaran



#### Subprogram Pemasukan/Pengeluaran

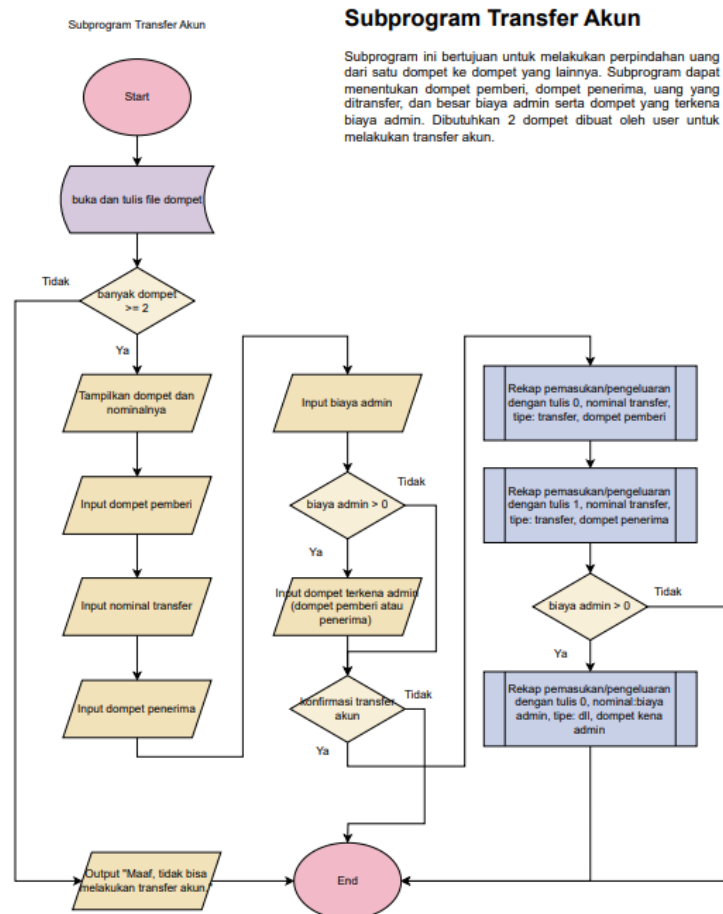
Subprogram ini bertujuan untuk melakukan pencatatan pemasukan/pengeluaran dari pengguna, baik itu pemasukan atau pengeluaran itu sendiri, nominalnya, tipe dari transaksi ini (gaji, uang saku, jajan, internet, dll) dan dompet yang dipakai. Dompet akan mencatat sejarah transaksi dan perubahan dompet pada file csv terkait.



Subprogram pemasukan dan pengeluaran berfungsi untuk melakukan pencatatan pemasukan/pengeluaran dari pengguna, baik itu nominalnya, tipe transaksi, dan dompet yang dipakai. Pada bagian ini, fungsi rekap pemasukan/pengeluaran dipakai untuk melakukan perubahan nominal pada dompet dan melakukan pencatatan transaksi di riwayat. Alur dari subprogram pemasukan dan pengeluaran adalah sebagai berikut:

1. Program dimulai dan pengguna memilih dompet untuk transaksi
2. Pengguna input 0 atau 1, dimana jika pengguna menginput 0 maka pengguna harus memasukkan jumlah nominal untuk pengeluaran, dan jika pengguna menginput 1 maka pengguna harus memasukkan jumlah nominal untuk pemasukan. Apabila pengguna menginput selain 0 atau 1 maka program akan kembali ke pilih dompet.
3. Setelah menginput nominal, pengguna menginput tipe pengeluaran/pemasukan yang diinginkan.
4. Kemudian setelah menginput tipe, melakukan finalisasi transaksi dan transaksi tersebut akan masuk ke rekap pemasukan/pengeluaran. Jika tidak akan ada pilihan untuk membuat transaksi baru atau tidak.
5. Jika user memilih untuk membuat transaksi baru maka program akan kembali ke halaman awal yaitu pilih dompet.

## 2.4 Transfer Akun



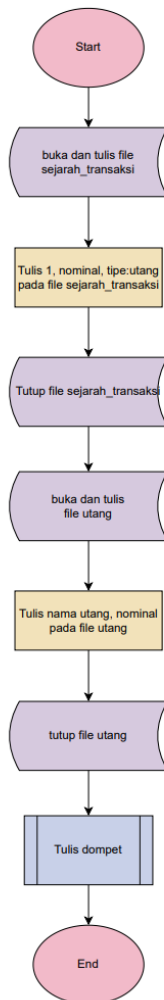
Subprogram transfer akun berfungsi untuk melakukan perpindahan sejumlah uang atau aset dari satu dompet ke dompet lainnya. Subprogram ini meminta input dompet pemberi, nominal transfer, dan dompet penerima sebagai bagian dari proses transfer. Pengguna juga akan diminta untuk input biaya admin dan dompet terkena admin (jika ada), alur dari subprogram transfer akun adalah sebagai berikut:

1. Saat subprogram dijalankan, membuka file dompet dan membaca isinya.
2. Apabila banyak dompet lebih dari atau sama dengan 2, subprogram akan dilanjutkan dan menanyakan input dompet pemberi, nominal transfer, dan dompet penerima kepada pengguna. Sedangkan jika banyak dompet kurang dari 2, akan muncul pesan bahwa transfer akun tidak bisa dilakukan dan mengakhiri subprogram.
3. Pengguna akan diminta input biaya admin, jika biaya admin yang diinput lebih dari 0, pengguna akan diminta input siapa dompet yang terkena biaya admin (antara dompet pemberi atau penerima).

Pengguna akan diminta konfirmasi dari input sebelumnya, apabila input diterima oleh pengguna, subprogram akan dilanjutkan pada serangkaian rekap pemasukan/pengeluaran untuk mencatat semua pemasukan dan pengeluaran pada dompet-dompet yang bersangkutan. Jika input tidak diterima oleh pengguna, subprogram diakhiri.

## 2.5 Buat Utang

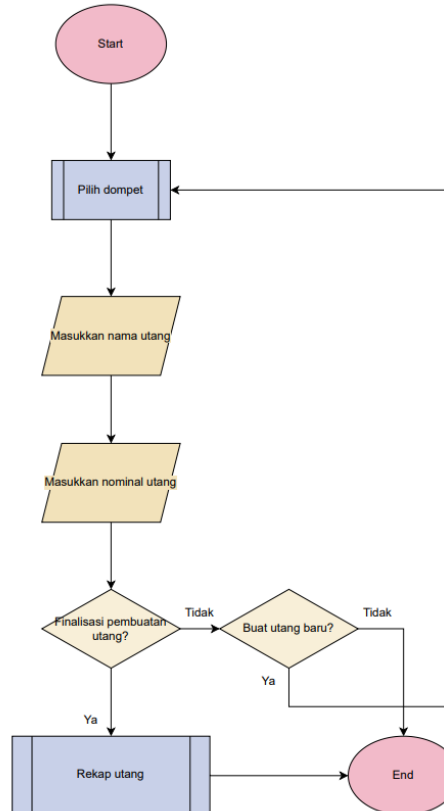
Subprogram Rekap Utang



Subprogram Buat Utang

Subprogram ini bertujuan untuk melakukan pencatatan utang dari pengguna berdasarkan nominalnya dan nama utang. Kemudian subprogram akan mencatat utang sebagai pendapatan (pada dompet dipilih) dan menambahkan utang pada csv terkait.

Subprogram Buat Utang



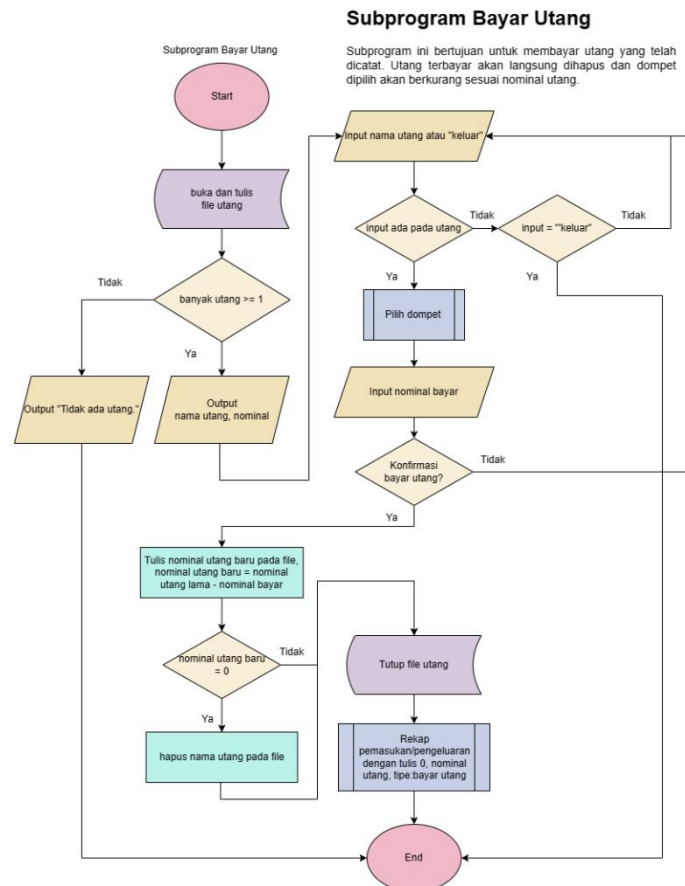
Subprogram buat utang berfungsi sebagai pencatatan utang dari pengguna berdasarkan nominalnya dan nama utang. Pada subprogram buat utang, pengguna diminta untuk memilih dompet, memasukkan nama utang, dan memasukkan nominal utang, dan mencatat utang itu jika diperlukan. Alur dari subprogram buat utang adalah sebagai berikut:

1. Program dimulai.
2. Pengguna diminta untuk memilih dompet.
3. Pengguna diminta untuk memasukkan nama utang dan nominal utang.
4. Program mengajukan pertanyaan kepada pengguna apakah ingin melakukan finalisasi pembuatan utang atau tidak dengan opsi “ya/tidak”.
5. Jika pengguna memilih “ya” untuk finalisasi pembuatan utang maka program akan merekap utang lalu program berakhir.



6. Jika pengguna memilih “tidak” untuk finalisasi pembuatan utang maka program akan mengajukan pertanyaan apakah ingin membuat utang baru atau tidak dengan opsi “ya/tidak”. Jika pengguna memilih “ya” untuk membuat utang baru maka program akan kembali ke bagian awal subprogram. Jika pengguna memilih “tidak” untuk membuat utang baru maka program berakhir.

## 2.6 Bayar Utang

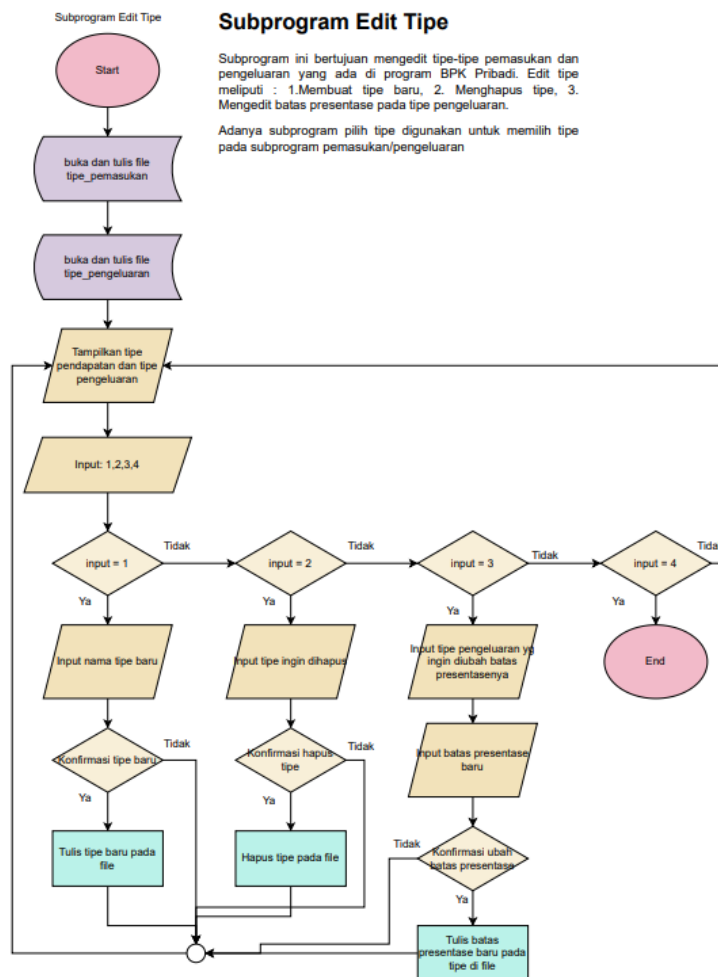


Subprogram bayar utang berfungsi untuk mencatat pembayaran utang dan menghapus catatan utang yang ada. Alur dari subprogram bayar utang adalah sebagai berikut:

1. Program dimulai.
2. Pengguna membuka atau menulis file utang.
3. Pengguna diminta untuk memasukkan angka untuk utang yang akan dibayar atau “tidak”.
4. Jika pengguna memilih “ya” pengguna diminta untuk memilih utang mana yang mau dibayar atau ingin “keluar” dari program.
5. Pengguna diminta untuk memilih dompet mana yang digunakan untuk membayar utang.

6. Pengguna diminta untuk memasukkan nominal untuk membayar utang jika memilih opsi “pilih dompet”.
7. Program akan meminta pengguna untuk konfirmasi apakah “ya” membayar utang atau “tidak”.
8. Jika pengguna memilih “ya” maka program akan mengurangi utang lama dengan nominal bayar yang pengguna masukkan tadi.
9. Program akan menghapus nama utang pada file kemudian akan merekap kembali utang dan program berakhir.
10. Jika pengguna memilih “tidak” maka program akan langsung berakhir.
11. Jika pengguna memilih “tidak” pada tahap 3 maka program akan langsung berakhir.

## 2.7 Edit Tipe

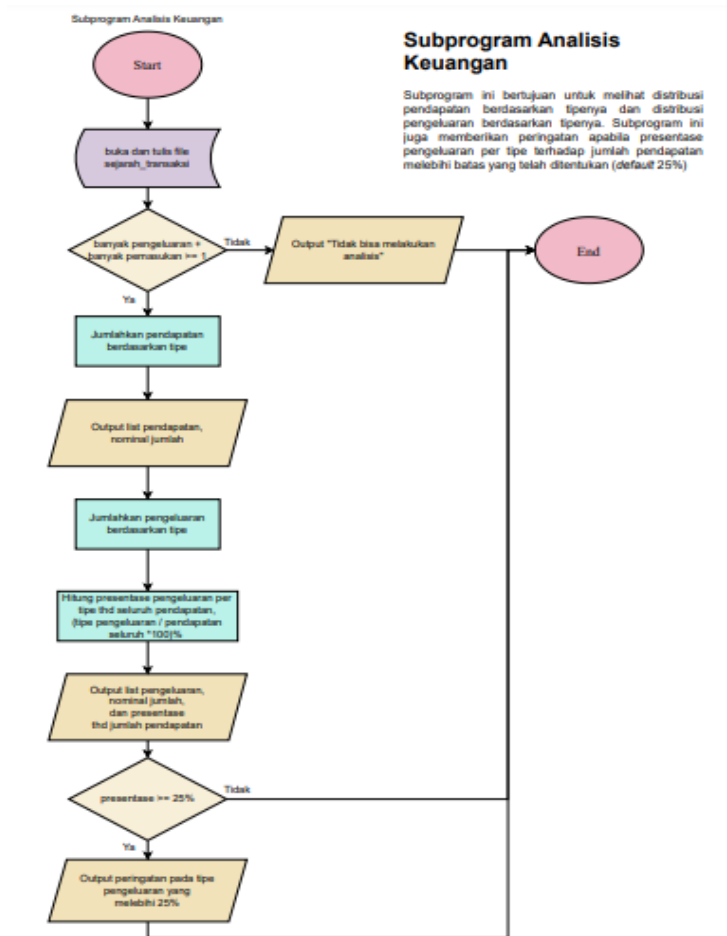


Subprogram edit tipe ini bertujuan untuk mengedit tipe-tipe pemasukan dan pengeluaran yang ada pada program. Subprogram edit tipe terdiri dari: 1. Membuatn tipe baru,

menghapus tipe, dan 3. Mengganti batas presentase bagi tipe pengeluaran. Alur dari subprogram edit tipe adalah sebagai berikut :

1. Saat subprogram dijalankan, membuka file tipe\_pemasukan dan tipe\_pengeluaran kemudian membaca isinya.
2. Tampilkan tipe pendapatan dan tipe pengeluaran, kemudian meminta input pengguna bilangan 1,2,3,4.
  - 2.1 Jika input adalah 1, meminta input nama baru kepada pengguna, kemudian konfirmasi input. Apabila input diterima, tulis tipe baru pada file dan kembali ke. Jika tidak, langsung kembali ke input bilangan 1,2,3,4.
  - 2.2 Jika input adalah 2, meminta input nama tipe untuk dihapus kepada pengguna, kemudian konfirmasi input. Apabila input diterima, tulis tipe baru pada file dan kembali ke. Jika tidak, langsung kembali ke input bilangan 1,2,3,4.
  - 2.3 Jika input adalah 3, meminta input nama tipe pengeluaran untuk diganti batas presentasinya kepada pengguna dilanjutkan input batas presentase baru, kemudian konfirmasi input. Apabila input diterima, tulis tipe baru pada file dan kembali ke. Jika tidak, langsung kembali ke input bilangan 1,2,3,4.
  - 2.4 Input 4 akan mengakhiri subprogram.

## 2.8 Analisis Keuangan

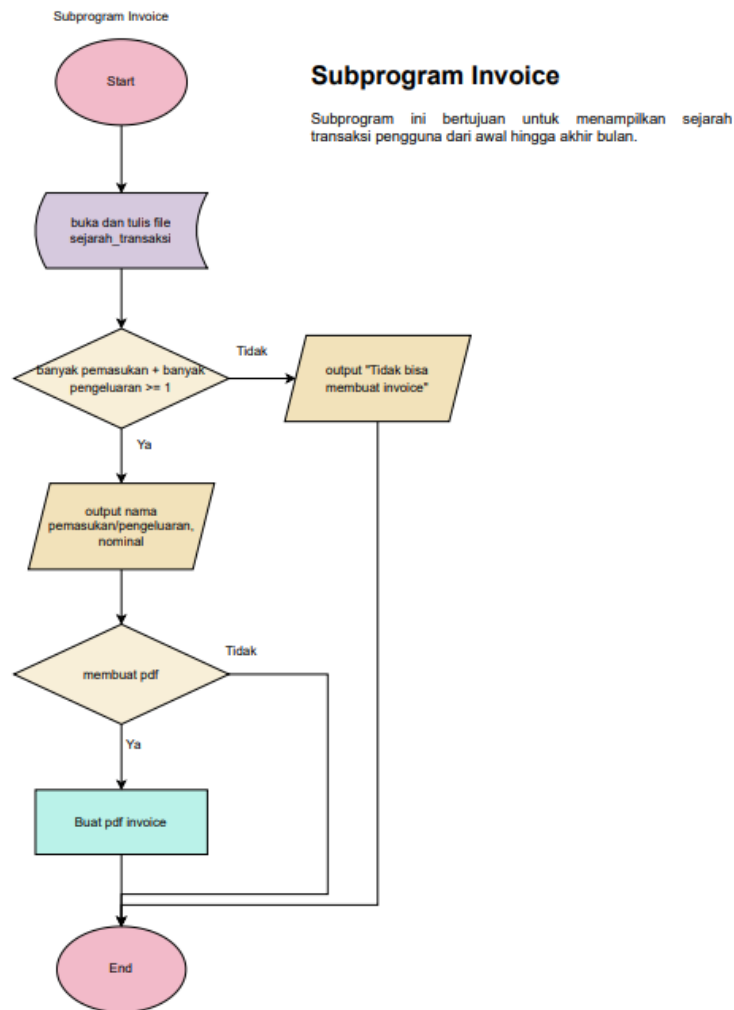


Subprogram analisis keuangan berfungsi untuk melihat distribusi pendapatan berdasarkan tipenya dan distribusi pengeluaran berdasarkan tipenya. Subprogram analisis keuangan membuka dan menulis file transaksi. Jika terdapat banyak pengeluaran dan banyak pemasukan dengan total minimal 1, program akan menghitung jumlah pendapatan dan pengeluaran berdasarkan tipe. Kemudian, program akan mencari persentase pengeluaran terhadap pendapatan total. Jika persentase pengeluaran  $\geq 25\%$ , program akan memberikan peringatan pada tipe pengeluaran yang melebihi batas tersebut. Jika tidak, program berakhir. Jika tidak ada transaksi yang memenuhi syarat, program akan mengeluarkan pesan "Tidak bisa melakukan analisis" dan berakhir.. Alur dari subprogram analisis keuangan adalah sebagai berikut :

1. Program dimulai.
2. File "sejarah\_transaksi" dibuka dan ditulis.

3. Program mengecek apakah jumlah pengeluaran dan jumlah pemasukan lebih besar atau sama dengan 1.
4. Jika ya, maka program melanjutkan ke langkah 5. Jika tidak, program melanjutkan ke langkah 11.
5. Program menghitung jumlah pendapatan berdasarkan tipe.
6. Program menampilkan daftar pendapatan beserta nominal jumlahnya.
7. Program menghitung jumlah pengeluaran berdasarkan tipe.
8. Program menghitung presentase pengeluaran terhadap seluruh pendapatan ( $\text{tipe pengeluaran} / \text{seluruh pendapatan} * 100\%$ ).
9. Program menampilkan daftar pengeluaran beserta nominal jumlahnya dan presentase terhadap jumlah pendapatan.
10. Program mengecek apakah presentase pengeluaran lebih besar atau sama dengan 25%.
11. Program menampilkan pesan "Tidak bisa melakukan analisis", lalu program berakhir.
12. Lanjutan dari langkah 10, jika ya, maka program menampilkan peringatan pada tipe pengeluaran yang melebihi 25%, kemudian program berakhir. Jika tidak, maka program berakhir langsung berakhir.

## 2.9 Invoice



Subprogram invoice berfungsi untuk menampilkan sejarah transaksi pengguna dalam rentang waktu sebulan. Subprogram membuka file "sejarah\_transaksi" untuk menyimpan riwayat transaksi. Pengguna diminta memilih antara "ya" atau "tidak". Jika "tidak", program berakhir. Jika "ya", program menampilkan nama pemasukan/pengeluaran dan nominalnya. Selanjutnya, program menanyakan apakah pengguna ingin membuat invoice PDF. Jika "tidak", program berakhir. Jika "ya", program membuat PDF invoice dan otomatis berakhir. Alur ini memberikan kemudahan dalam melihat riwayat transaksi dan membuat invoice PDF sesuai kebutuhan pengguna. Alur dari subprogram invoice adalah sebagai berikut :

1. Program dimulai
2. Program akan membuka dan tulis file "sejarah\_transaksi".
3. Pengguna diminta memasukkan "ya" atau "tidak".

4. Jika pengguna memilih "tidak" maka program akan langsung berakhir.
5. Jika pengguna memilih "ya" maka program akan menampilkan nama pemasukan/pengeluaran serta nominal.
6. Program akan menanyakan apakah pengguna ingin membuat pdf dari invoice atau "tidak".
7. Jika pengguna memilih "tidak" maka program akan langsung berakhir
8. Jika pengguna memilih "ya" maka program akan langsung membuatkan pdf dan akan langsung mengakhiri program.

## BAB III

### KODE PROGRAM

#### 3.1 Main Program

```

main.py > ...
1  from pemasukan_pengeluaran import pemasukan_pengeluaran
2  from transfer_akun import transfer_akun
3  from buat_utang import buat_utang
4  from bayar_utang import bayar_utang
5  from edit_tipe import edit_tipe
6  from analisis_keuangan import analisis_keuangan
7  from invoice import invoice
8  from menampilkan_dompot_utang import menampilkan_dompot_utang
9  import submodules
10 from os import system
11 from time import sleep
12

```

Bagian import pada program main.

```

13 def main():
14     while True:
15         # Bersihkan Layar
16         system("cls")
17
18         # Tampilkan identitas program
19         print()
20         print(" SELAMAT DATANG DI PROGRAM ".center(50,"="))
21         print(" BUDGETING DAN PENCATATAN ".center(50,"="))
22         print(" KEUANGAN PRIBADI ".center(50,"="))
23         print()
24
25         # Tampilkan dompet dan utang
26         menampilkan_dompot_utang()
27         print()
28
29         # Opsi menu
30         menu = [
31             [1, "Pemasukan / Pengeluaran", pemasukan_pengeluaran],
32             [2, "Transfer Akun", transfer_akun],
33             [3, "Buat Utang", buat_utang],
34             [4, "Bayar Utang", bayar_utang],
35             [5, "Edit Tipe", edit_tipe],
36             [6, "Analisis Keuangan", analisis_keuangan],
37             [7, "Invoice", invoice],
38             [0, "Keluar", None]
39         ]
40         opsi = list(range(8))

```

Bagian identitas program dan pembuatan menu pada program main.

```

42 # Tampilkan menu
43 submodules.display_table([ [ ele[0], ele[1] ] for ele in menu ], ["", "Menu"])
44 print()
45 pilih_menu = submodules.input_of_int_options("Input 1-7 untuk pilih menu atau 0 untuk keluar ", opsi)
46
47 if pilih_menu != 0:
48     system("cls")
49     menu[pilih_menu-1][2]()
50     _ = submodules.input_normal(submodules.ch_color_style("Tekan Enter untuk kembali ke menu utama", "yellow"))
51     print("... Proses")
52     sleep(.75)
53     print("... Kembali ke")
54     sleep(.75)
55     print("... Menu utama")
56     sleep(.75)
57     print("... -----")
58     sleep(.75)
59
60 else:
61     konfir_keluar = submodules.input_of_yatidak("Anda yakin mau keluar? (y/t)")
62     if konfir_keluar == "y":
63         print(submodules.ch_color_style("JUMPA LAGI, SEMANGAT BERHEMAT", "orange"))
64         break
65
66 if __name__ == "__main__":
67     main()

```

Bagian tampilan menu, pengambilan input, dan pemanggilan subprogram pada program main.



## 3.2 pdf\_1

```
pdf_1.py > buat_pdf_1
1 import submodules
2 from datetime import date, timedelta
3 from reportlab.lib.pagesizes import letter
4 from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph
5 from reportlab.lib.styles import getSampleStyleSheet
6 from reportlab.lib import colors
```

Bagian import pada subprogram pdf\_1.

```
7
8 def buat_pdf_1():
9     hd_iv = ["", "Tanggal", "Dompet", "Tipe", "Pemasukan", "Pengeluaran"]
10    ls_iv = []
11
12    _, ls_tr = submodules.open_read_csv("sejarah_transaksi.csv")
13
14    # Restriksi 30 hari yg lalu
15    hari_ini = date.today()
16    hari_30_belakang = (hari_ini - timedelta(days=30)).strftime("%Y/%m/%d")
17    ls_tr_new = []
18    for ele in ls_tr:
19        if ele[0] < hari_30_belakang:
20            break
21        ls_tr_new.append(ele)
22
23    # Pembuatan baris invoice pertabel
24    for id, ele in enumerate(ls_tr_new):
25        if ele[1] == "1":
26            ls_iv.append([id+1, ele[0], ele[3], ele[2], f"Rp{int(ele[4]):>12,}", None])
27        else:
28            ls_iv.append([id+1, ele[0], ele[3], ele[2], None, f"Rp{int(ele[4]):>12,}" ])
29
30    # Pembuatan file pdf
31    output_file = "pdf_1.pdf"
32    title_text = "Tabel sejarah transaksi 30 hari terakhir"
33    doc = SimpleDocTemplate(output_file, pagesize=letter, orientation='portrait')
34    elements = []
```

Bagian pembuatan list tanggal, list transaksi, dan persiapan pembuatan file pada subprogram pdf\_1.

```
pdf_1.py > buat_pdf_1
35
36    # Penambahan judul pada file
37    styles = getSampleStyleSheet()
38    title = Paragraph(title_text, styles['Title'])
39    elements.append(title)
40
41    # Mengumpulkan data tabel terdiri dari header dan isi
42    table_data = [hd_iv] + ls_iv
43
44    # Membuat objek/memori tabel
45    table = Table(table_data)
46
47    # Membuat model tabel
48    style = TableStyle([
49        ('ALIGN', (0, 0), (-1, -1), 'RIGHT'),
50        ('TEXTCOLOR', (0, 0), (-1, 0), colors.black), # Setting font color header
51        ('BACKGROUND', (0, 0), (-1, 0), colors.grey), # Setting background header
52        ('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'), # Setting font style header
53        ('FOUNTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'), # Setting font style header
54        ('BOTTOMPADDING', (0, 0), (-1, 0), 8),
55        ('TOPPADDING', (0, 0), (-1, 0), 8),
56        ('BOTTOMPADDING', (0, 1), (-1, -1), 4),
57        ('TOPPADDING', (0, 1), (-1, -1), 4),
58        ('GRID', (0, 0), (-1, -1), 1, colors.black), # Setting cell border
59        ('BACKGROUND', (4, 1), (4, -1), colors.lightgreen), # Setting background hijau kolom ke-4
60        ('BACKGROUND', (5, 1), (5, -1), colors.lightcoral), # Setting background merah kolom ke-5
61    ])
62
63    table.setStyle(style)
64
65    # Tambahkan tabel
66    elements.append(table)
67
68    # Buat file pdf
69    doc.build(elements)
```

Bagian pembuatan tabel dan pembuatan file pdf pada subprogram pdf\_1.

## 3.3 pdf\_2\_3

```

pdf_2_3.py > buat_pdf_2_3
1 import matplotlib.pyplot as plt
2 from reportlab.pdfgen import canvas
3 import datetime
4 import submodules
5
6 def buat_pdf_2_3():
7     # dapatkan tanggal hari ini
8     current_date = datetime.date.today()
9
10    # list tanggal 30 hari ke belakang
11    date_list = []
12
13    # Restriksi 30 hari ke belakang
14    for i in range(30, -1, -1):
15        date = current_date - datetime.timedelta(days=i)
16        date_list.append(date.strftime("%Y/%m/%d"))
17
18    # Pembuatan nilai tren pemasukan-pengeluaran
19    ls_tr = submodules.open_read_csv("sejarah_transaksi.csv")
20    money_flow = []
21    for date in date_list:
22        money_flow_on_date = 0
23        for baris in ls_tr:
24            date_str, kode, _, nominal = baris
25            kode, nominal = int(kode), int(nominal)
26            if date == date_str:
27                money_flow_on_date = money_flow_on_date + nominal if kode == 1 else money_flow_on_date - nominal
28            elif date > date_str:
29                break

```

Bagian import, pembuatan list tanggal, dan pembuatan list tren pemasukan-pengeluaran pada subprogram pdf\_2\_3.

```

pdf_2_3.py > buat_pdf_2_3 > create_graph
32 # Pembuatan nilai perubahan aset
33 _, ls_dp = submodules.open_read_csv("dompet.csv")
34 sum_dp = sum([int(ele[1]) for ele in ls_dp])
35 asset_flow = []
36 asset_on_date = 0
37 for id, money in enumerate(money_flow[::-1]):
38     if id == 0:
39         asset_on_date = sum_dp
40     else:
41         asset_on_date -= money_flow[::-1][id-1]
42         asset_flow.insert(0, asset_on_date)
43
44 # Bagian pembuatan grafik dan file pdf
45 def create_graph(fig, y_value, nama, legenda):
46     # Pengambilan value x dan value y
47     x = [ele[-5:] for ele in date_list]
48     y = y_value
49     y_line = 0
50
51     # Pembuatan garis fungsi dan titik nilai
52     colors = ['green' if yi > y_line else 'red' for yi in y]
53
54     plt.figure(fig, figsize=(10, 8))
55     plt.plot(x, y, color="gray")
56     plt.scatter(x, y, color=colors)
57
58     # Pembuatan garis y = 0
59     plt.axhline(y=y_line, color='black', linestyle='-')

```

Bagian pembuatan list perubahan aset dan bagian pembuatan grafik pada subprogram pdf\_2\_3.

```

60
61 # Label-Label pada grafik
62 plt.xlabel('Tanggal')
63 plt.ylabel(legenda)
64 plt.xticks(rotation='vertical')
65 plt.title(nama)
66 plt.grid(True)
67 plt.legend([legenda])
68
69 plt.ticklabel_format(style='plain', axis='y')
70
71 # Pembuatan file pdf
72 pdf_file = f'pdf_{fig}.pdf'
73
74 plt.savefig(f'pdf_{fig}.png', dpi=500)
75
76 c = canvas.Canvas(pdf_file)
77 c.setPageSize((800, 600))
78 c.drawImage(f'pdf_{fig}.png', 60, 60, width=700, height=500)
79 c.save()

```

Bagian pembuatan grafik dan pembuatan pdf pada subprogram pdf\_2\_3.

```

81 # Pembuatan grafik tren pemasukan-pengeluaran
82 create_graph(2, money_flow.copy(), "Tren pemasukan-pengeluaran 30 hari terakhir", "Tren pemasukan-pengeluaran")
83 # Pembuatan grafik perubahan aset
84 create_graph(3, asset_flow.copy(), "Tren perubahan aset 30 hari terakhir", "perubahan aset")
85

```

Bagian pemanggilan fungsi pembuat grafik untuk tren pemasukan-pengeluaran dan perubahan aset pada subprogram pdf\_2\_3.

### 3.4 pdf\_4\_5

```

pdf_4_5.py > buat_pdf_4_5
1 import matplotlib.pyplot as plt
2 import submodules
3
4 def buat_pdf_4_5():
5     # Pembacaan file sejarah transaksi
6     _, ls_tr = submodules.open_read_csv("sejarah_transaksi.csv")
7
8     # Persiapan pembuatan x-y key-value untuk distribusi pendapatan
9     dict_ty1 = {}
10    # Persiapan pembuatan x-y key-value untuk distribusi pengeluaran
11    dict_ty0 = {}
12
13    # pengisian x-y
14    for ele in ls_tr:
15        _, kode, tipe, _, nominal = ele
16        # Distribusi tidak melibatkan tipe transfer
17        if tipe == "transfer_akun":
18            continue
19        if kode == "1":
20            if tipe not in dict_ty1:
21                dict_ty1[tipe] = int(nominal)
22            else:
23                dict_ty1[tipe] += int(nominal)
24        else:
25            if tipe not in dict_ty0:
26                dict_ty0[tipe] = int(nominal)
27            else:
28                dict_ty0[tipe] += int(nominal)
29
30    # Label dan ukuran untuk distribusi pengeluaran
31    label0 = dict_ty0.keys()
32    sizes0 = dict_ty0.values()
33
34    # Label dan ukuran untuk distribusi pendapatan
35    label1 = dict_ty1.keys()
36    sizes1 = dict_ty1.values()
37

```

Bagian import dan pembuatan *key-value* nama tipe dan ukuran transaksi pada subprogram pdf\_4\_5.

```

38 | # Pembuatan pie chart
39 | def create_chart(fig, dis, values, file, nama):
40 |     total = sum(values)
41 |
42 |     plt.figure(fig)
43 |     plt.pie(values, labels=dis, autopct=lambda pct: f"{pct:.2f}% (Rp{round(pct * total / 100,0):,})", wedgeprops={'edgecolor': 'white'})
44 |
45 |     plt.axis('equal')
46 |
47 |     center_circle = plt.Circle((0, 0), 0.5, color='white')
48 |     plt.gca().add_artist(center_circle)
49 |
50 |     plt.text(.98, .96, f"Total: Rp{total:,}", ha='center')
51 |
52 |     plt.title(nama)
53 |
54 |     # plt.show()
55 |     plt.savefig(file+'.pdf')
56 |
57 | # Pembuatan pie chart untuk distribusi pemasukan
58 | create_chart(4, label1, sizes1, "pdf_4", "Distribusi pemasukan dalam 30 hari terakhir")
59 | # Pembuatan pie chart untuk distribusi pengeluaran
60 | create_chart(5, label0, sizes0, "pdf_5", "Distribusi pengeluaran dalam 30 hari terakhir")
61 |
62 | if __name__ == "__main__":
63 |     buat_pdf_4_5()

```

Bagian pembuatan *chart* dan pemanggilan fungsi pembuatan *chart* untuk distribusi pendapatan dan distribusi pengeluaran pada subprogram pdf\_4\_5.

### 3.5 Submodules

```

submodules.py > ...
1  from os import getcwd, path
2  from csv import reader, writer
3  from tabulate import tabulate
4

```

Bagian import file submodules.

```

def open_read_csv(filename):
    filepath = path.join(getcwd(), "files", filename)
    with open(filepath, "r", newline="") as f:
        lines = filter(lambda x: x.strip(), f) # Exclude empty lines
        baca = list(reader(lines))
        header = baca[0]
        content = baca[1:]

    return header, content

```

Fungsi open\_read\_csv pada submodules untuk membaca file.

```

def open_write_all_csv(filename, ds, hd):
    filepath = path.join(getcwd(), "files", filename)
    with open(filepath, "w", newline="") as f:
        tulis = writer(f)
        tulis.writerow(hd)
        tulis.writerows(ds)

```

Fungsi open\_write\_all\_csv pada submodules untuk menulis semua file.

```
def open_append_csv(filename, ds):
    filepath = path.join(getcwd(), "files", filename)
    with open(filepath, "a", newline="") as f:
        tulis = writer(f)
        tulis.writerows(ds)
```

Fungsi open\_append\_csv pada submodules untuk menambahkan file terakhir.

```
def open_append2first_csv(filename, ds):
    header, content = open_read_csv(filename)
    toinsert = ds[0]
    content.insert(0, toinsert)
    open_write_all_csv(filename, content, header)
```

Fungsi open\_append2first\_csv pada submodules untuk menambahkan file paling awal.

```
def input_normal(prompt:str):
    while True:
        try:
            ans = input(prompt+"\n")
            break
        except AssertionError as er:
            print(er)
    return ans.lower()
```

Fungsi input\_normal pada submodules untuk menginput tanpa syarat.

```
def input_of_int_options(prompt:str, ls_options:list, errmsg = "Input Tidak Valid!"):
    while True:
        try:
            ans = int(input(prompt+"\n"))
            assert ans in ls_options, errmsg
            break
        except AssertionError as er:
            print(er)
        except ValueError:
            print(errmsg, f"\nmasukkan opsi yang valid: {ls_options}")
    return ans
```

Fungsi input\_of\_int\_options pada submodules untuk menginput dengan syarat.

```
def input_of_yatidak(prompt:str, errmsg= "Input Tidak Valid!"):
    while True:
        try:
            ans = input(prompt+"\n")
            assert ans.lower() in ["y", "t"], errmsg
            break
        except AssertionError as er:
            print(er)
    return ans.lower()
```

Fungsi input\_of\_yatidak pada submodules untuk menginput dengan syarat y atau t.

```
def input_money(prompt:str):
    while True:
        try:
            amount = int(input(prompt+"\n"))
            assert amount >= 0, "Uang tidak negatif!"
            break
        except AssertionError as er:
            print(er)
        except ValueError:
            print("Masukkan uang dengan benar")
    return amount
```

Fungsi input\_money pada submodules untuk menginput nominal uang.

```
def input_money_w_params(prompt:str, code:int, moneyparam:int):
    counter = 0
    while True:
        try:
            if counter >= 3:
                return None
            amount = int(input(prompt+"\n"))
            assert amount >= 0, "Uang tidak negatif!"
            counter += 1
            assert moneyparam - amount >= 0 if code == 0 else True, "Uang tidak cukup!"
            break
        except ValueError:
            print("Masukkan uang dengan benar")
        except AssertionError as er:
            print(er)
    return amount
```

Fungsi input\_money\_w\_params pada submodules untuk menginput nominal uang dengan syarat.

```
def ch_color_style(value, color:str="", style:str=""):
    color, style = color.lower(), style.lower()
    fmt = {"reset" : "\033[0m",
           "bold" : "\033[1m",
           "italic" : "\033[3m",
           "underline" : "\033[4m",
           "" : "",
           "black" : "\033[30m",
           "red" : "\033[38;5;196m",
           "orange" : "\033[38;2;255;165;55m",
           "yellow" : "\033[38;2;255;255;0m",
           "green" : "\033[38;5;46m",
           "blue" : "\033[34m",
           "magenta" : "\033[35m",
           "purple" : "\033[38;5;165m",
           "sky" : "\033[38;5;111m",
           "tosca" : "\033[38;5;123m",
           "white" : "\033[37m"}
    return f"{fmt[style]}{fmt[color]}{value}{fmt['reset']}"
```

Fungsi ch\_color\_style pada submodules untuk membuat warna pada tulisan dan table.

```
def display_table(values:list, header:list, header_cond:bool= True):
    print(tabulate(values, headers= header if header_cond == True else False, tablefmt="rst", stralign= "center"))
```

Fungsi display\_table pada submodules untuk menampilkan tabel tertentu.

### 3.6 Tranfer\_akun

```
from . import submodules as sdl
from .pemasukan_pengeluaran import rekap_pemasukan_pengeluaran
```

Import submodules dan file pada subprogram transfer\_akun

```
def transfer_akun():
    # Akses file dompet
    hd_dompet, ls_dompet = sdl.open_read_csv("dompet.csv")
    byk_dompet = len(ls_dompet)

    # Identitas Subprogram
    print("\n"+" TRANSFER AKUN ".center(50,"=")+"\n")
```

Bagian mengakses file dompet dan mencetak identitas subprogram

```
# Tampilkan dompet
dis_tl = [ [id+1, row[0], f"Rp{int(row[1]):^12,}"] for id,row in enumerate(ls_dompet) ]
sdl.display_table(dis_tl, [""] + hd_dompet)

opsi = list(range(1,byk_dompet+1))
lbl_1 = sdl.input_of_int_options(f"Input 1-{byk_dompet} untuk memilih dompet pemberi transfer ", opsi)
dompet_1, nominal_1 = ls_dompet[lbl_1-1][0], int(ls_dompet[lbl_1-1][1]) # Ambil dompet 1

nominal_tf = sdl.input_money_w_params(f"Masukkan jumlah uang ditransfer ",0, nominal_1)
if nominal_tf == None:
    print(sdl.ch_color_style("Uang tidak cukup, dialihkan kembali ke menu utama","yellow"))
    return # Ketika nominal pengeluaran gagal didapatkan

opsi.remove(lbl_1) # Hapus pilihan dompet satu untuk persiapan input dompet kedua
lbl_2 = sdl.input_of_int_options(f"Input 1-{byk_dompet} untuk memilih dompet penerima transfer ", opsi, f"Pilih dompet yang ada kecuali dompet {dompet_1}!")
dompet_2, nominal_2 = ls_dompet[lbl_2-1][0], int(ls_dompet[lbl_2-1][1]) # ambil dompet 2

nominal_1 -= nominal_tf # Pengurangan nominal 1 untuk persiapan pengurangan dompet
```

Bagian menampilkan dompet

```
if adm > 0:
    # jika biaya admin ada
    print("Opsi dompet terkena admin")
    print(f"1. {dompet_1}")
    print(f"2. {dompet_2}")
    count = 0
    while True:
        try:
            if count >= 3:
                print(sdl.ch_color_style("Uang tidak cukup, dialihkan kembali ke menu utama","yellow"))
                return # Ketika tidak didapatkan siapa dompet kena admin
            kena_admin = sdl.input_of_int_options("Dompet mana yang terkena admin? ", [1,2])
            count += 1
            assert nominal_1 - adm >= 0 if kena_admin == 1 else nominal_2 - adm >= 0 if kena_admin == 2 else True, "Uang tidak cukup!"
            break
        except AssertionError as er:
            print(er)
    else:
        kena_admin = 0 # Tidak ada biaya admin
```

```
nominal_dis = f"Rp{nominal_tf:,}"
print(f'''Konfirmasi transfer akun :
Transfer sebanyak {sdl.ch_color_style(nominal_dis, "sky")}
Dari {sdl.ch_color_style(domp1, "sky")}
ke {sdl.ch_color_style(domp2, "sky")}''')

if adm > 0:
    if kena_adm == 1:
        print(f'Biaya admin {sdl.ch_color_style(f"Rp{adm:,}", "red")} pada {dompet_1}')
    elif kena_adm == 2:
        print(f'Biaya admin {sdl.ch_color_style(f"Rp{adm:,}", "red")} pada {dompet_2}')

konfirmasi = sdl.input_of_yatidak(f"Yakin melakukan transfer akun? (y/t) ")
if konfirmasi == "y":
    rekap_pemasukan_pengeluaran(0, "transfer_akun", dompet_1, nominal_tf)
    rekap_pemasukan_pengeluaran(1, "transfer_akun", dompet_2, nominal_tf)

    if kena_adm == 1:
        rekap_pemasukan_pengeluaran(0, "dan lain-lain", dompet_1, adm)
    elif kena_adm == 2:
        rekap_pemasukan_pengeluaran(0, "dan lain-lain", dompet_2, adm)

if __name__ == "__main__":
    transfer_akun()
```

Bagian konfirmasi biaya admin dan konfirmasi transfer akun

### 3.7 Buat\_utang

```
from . import submodules as sdl
from .pilih_dompet import pilih_dompet
from .pemasukan_pengeluaran import rekap_pemasukan_pengeluaran
```

Import submodules dan import file pada subprogram buat\_utang



```
def rekap_utang(nama:str, dompet:int, nominal:int):
    # Akses tanggal pembuatan transaksi (hari ini)
    rekap_pemasukan_pengeluaran(1, "utang", dompet, nominal)

    # Persiapan list ditulis ke csv
    toadd = [[ nama, nominal ]]

    # Penulisan ke csv pada index value pertama
    sdl.open_append_csv("utang.csv", toadd)

def buat_utang():
    while True: # Pengulangan selama belum keluar dari subprogram
        # Identitas Subprogram
        print("\n"+"BUAT UTANG ".center(50,"=")+"\n")

        # pilih dompet untuk mencatat utang
        dompet, nominal_dompot = pilih_dompot()

        # meminta pengguna memasukkan nama utang
        nama_utang = sdl.input_normal("Masukkan nama utang: ")

        # meminta pengguna memasukkan nominal utang
        nominal_utang = sdl.input_money("Masukkan nominal utang: ")

        # Konfirmasi buat utang
        dis_nom = f"Rp{nominal_utang:,.2f}"
        print(f'''Konfirmasi pembuatan transaksi:
Transaksi {sdl.ch_color_style("pemasukan","sky")}
Tipe {sdl.ch_color_style("utang","sky")}
Nominal {sdl.ch_color_style(dis_nom,"sky")} pada dompet {sdl.ch_color_style(dompot,"sky")}
''')
```

Bagian mengakses tanggal pembuatan transaksi (hari ini), persiapan list ditulis pada csv, penulisan pada csv index value pertama, mencetak identitas subprogram, memilih dompet untuk mencatat utang, meminta pengguna memasukkan nama utang dan nominal utang, dan konfirmasi buat utang.

```
# memberikan pilihan kepada pengguna apakah ingin menyelesaikan pembuatan utang atau tidak
finalisasi = sdl.input_of_yatidak("Apakah Anda ingin finalisasi pembuatan utang? (y/t) ")

# jika pengguna ingin menyelesaikan pembuatan utang
if finalisasi == "y":
    # menambahkan data utang ke file CSV
    rekap_utang(nama_utang, dompet, nominal_utang)
    # keluar dari loop while
    break

# jika pengguna tidak ingin menyelesaikan pembuatan utang
elif finalisasi == "t":
    # meminta pengguna memilih apakah ingin membuat utang baru atau tidak
    buat_lagi = sdl.input_of_yatidak("Apakah Anda ingin membuat utang baru? (input t untuk keluar dari subprogram) (y/t): ")
    # jika pengguna tidak ingin membuat utang baru
    if buat_lagi == "t":
        # keluar dari loop while
        break

if __name__ == "__main__":
    buat_utang()
```

Bagian memberikan opsi kepada pengguna apakah ingin menyelesaikan pembuatan utang atau tidak, konfirmasi pembuatan utang, meminta pengguna memilih apakah ingin membuat utang baru atau tidak.

### 3.8 Bayar\_utang

```
import submodules
from pilih_dompot import pilih_dompot
from pemasukan_pengeluaran import rekap_pemasukan_pengeluaran
```

Import file pada subprogram bayar\_utang.

```
def bayar_utang():
    # Identitas subprogram
    print("\n" + "BAYAR UTANG".center(50, "=") + "\n")

    # Membuka file CSV dan mendapatkan header dan daftar utang
    header, list_utang = submodules.open_read_csv("utang.csv")
    banyak_utang = len(list_utang)

    # Memeriksa apakah ada utang dalam daftar
    if banyak_utang >= 1:
        # Menampilkan daftar utang
        dis_ls_utang = [ [id+1, row[0], f"Rp{int(row[1]):>10,}"] for id, row in enumerate(list_utang) ]
        dis_ls_utang.append([0, "keluar"])

        submodules.display_table(dis_ls_utang, [""] + header)

    # Meminta pengguna untuk memilih utang yang akan dibayar atau keluar dari program
    while True: # Pengulangan permintaan input pilihan utang
        # Meminta input user untuk pemilihan utang dibayar
        opsi = list(range(banyak_utang+1))
        if banyak_utang == 1:
            utang_bayar = submodules.input_of_int_options(f"Input 1 untuk utang dibayar atau 0 untuk keluar", opsi)
        else:
            utang_bayar = submodules.input_of_int_options(f"Input 1-{banyak_utang} untuk utang dibayar atau 0 untuk keluar", opsi)
```

Bagian identitas subprogram, membuka file csv, memeriksa utang, menampilkan daftar utang, meminta pengguna memilih utang yang akan dibayar pada fungsi bayar\_utang.

```
# Apabila user memilih keluar
if utang_bayar == 0:
    konfir_keluar = submodules.input_of_yatidak("Yakin keluar dari subprogram? (y/t) ")
    if konfir_keluar == "y":
        print("Keluar dari subprogram")
        return None # Keluar dari loop dan subprogram
    else:
        continue

# Apabila memilih utang dibayar
elif utang_bayar in opsi:
    nominal_utang = int(list_utang[utang_bayar-1][1]) # Ambil nominal utang terbayar

# Melanjutkan ke langkah selanjutnya: memilih dompet dan nominal pembayaran
dompet, nominaldompet = pilih_dompot() # Ambil dompet

# Input nominal bayar utang
while True:
    try:
        dibayar = submodules.input_money_w_params("Masukkan nominal bayar utang? ", 0, nominaldompet)
        assert nominal_utang - dibayar >= 0, f"Nominal bayar terlalu banyak, maksimal {nominal_utang}"
        break
    except AssertionError as er:
        print(er)

nominal_utang_dis = f"Rp{nominal_utang:,}"
dibayar_dis = f"Rp{dibayar:,}"
# Konfirmasi bayar utang
print(f'''Konfirmasi bayar utang:
```

Bagian ketika user memilih keluar, apabila pilih utang dibayar, memilih dompet dan input nominal bayar pada fungsi bayar\_utang.

```

nama utang = {submodules.ch_color_style(list_utang[utang_bayar-1][0],"sky")}
nominal utang = {submodules.ch_color_style(nominal_utang_dis,"sky")}
dibayar = {submodules.ch_color_style(dibayar_dis,"red")} dengan dompet {submodules.ch_color_style(dompet,"sky")}'''

    # Apabila konfir bayar utang, keluar dari while loop
    konfir_bayar = submodules.input_of_yatidak("Apakah mau membayar utang? (y/t) ")
    if konfir_bayar == "y":
        break
    # Apabila tidak konfir, kembali mengulang while loop

# Pengurangan utang
list_utang[utang_bayar-1][1] = nominal_utang - dibayar

# Apabila sisa utang 0, utang dihapus
if list_utang[utang_bayar-1][1] == 0:
    list_utang.pop(utang_bayar-1)

# Penulisan file utang
submodules.open_write_all_csv("utang.csv", list_utang, header)

# Penulisan rekap pemasukan/pengeluaran
rekap_pemasukan_pengeluaran(0, "bayar utang", dompet, dibayar)

# Apabila tidak ada utang
else:
    print('Tidak ada utang')

```

Bagian konfirmasi bayar utang, pengurangan utang, penulisan file utang, rekap pemasukan\pengeluaran pada fungsi bayar\_utang.

### 3.9 Edit\_tipe

```

import submodules

def buat_tipe(code:int):
    # Konfirmasi tipe yang dipilih sebelumnya
    if code == 1:
        tipe = "Pemasukan"
    else:
        tipe = "Pengeluaran"
    # Buat tipe baru apabila input 0
    nama = submodules.input_normal(f"Masukkan nama tipe {tipe}")

    # Konfirmasi tipe baru
    print(f'''Konfirmasi pembuatan tipe {tipe}
nama tipe = {submodules.ch_color_style(nama,"sky")}''')

    konfir_input = submodules.input_of_yatidak(f"Apakah mau menyimpan tipe {tipe}? (y/t) ")

    # Apabila input y, simpan tipe
    if konfir_input == "y":
        if code == 1:
            tulis = [[ nama ]]
            submodules.open_append_csv("tipe_pemasukan.csv", tulis)
        else:
            tulis = [[ nama, 25 ]]
            submodules.open_append_csv("tipe_pengeluaran.csv", tulis)

```

Bagian import submodules, konfirmasi tipe, buat tipe baru, konfirmasi tipe baru, dan dimpan tipe pada fungsi buat\_tipe.

```

26
27 def pilih_tipe(code:int):
28     while True:
29         # K (parameter) code: int      ilah sebelumnya
30         if code == 1:
31             hd, ls_tp = submodules.open_read_csv("tipe_pemasukan.csv")
32             dis_ls_tp = [ [id+1] + row for id,row in enumerate(ls_tp) ]
33             tipe = "Pemasukan"
34         else:
35             hd, ls_tp = submodules.open_read_csv("tipe_pengeluaran.csv")
36             dis_ls_tp = [ [id+1, row[0]] for id,row in enumerate(ls_tp) ]
37             tipe = "Pengeluaran"
38
39         # Tampilkan tipe yang ada
40         dis_ls_tp.append([0, "buat"])
41
42         print("Daftar Tipe dapat dipilih: ")
43         submodules.display_table(dis_ls_tp, [""]+hd)
44
45         # Input pilihan tipe
46         banyak_tipe = len(ls_tp)
47         opsi = list(range( banyak_tipe+1 ))
48
49         pilih = submodules.input_of_int_options(f"Input 1-{banyak_tipe} untuk pilih tipe {tipe} atau 0 untuk buat tipe ", opsi)
50
51         # Mengembalikan tipe terpilih
52         if pilih != 0:
53             return ls_tp[pilih-1][0]
54
55         # Apabila input 0 / buat tipe
56         buat_tipe(code)
57

```

Bagian konfirmasi tipe, tampilkan tipe,input pilih tipe, kemudian buat tipe pada fungsi pilih\_tipe.

```

def edit_tipe():
    while True: # Pengulangan subprogram kecuali user keluar
        # Identitas subprogram
        print("\n"+"EDIT TIPE".center(50,"=")+"\n")

        # Mengakses masing-masing tipe pendapatan dan pengeluaran
        hd_in, ls_in = submodules.open_read_csv("tipe_pemasukan.csv")
        hd_ex, ls_ex = submodules.open_read_csv("tipe_pengeluaran.csv")

        byk_tipe1 = len(ls_in)
        ls_in_dis = [ [id+1] + row for id,row in enumerate(ls_in) ]

        byk_tipe2 = len(ls_ex)
        ls_ex_dis = [ [id+1+byk_tipe1] + row for id,row in enumerate(ls_ex) ]

        # Menampilkan daftar tipe pemasukan
        print("Daftar Tipe Pemasukan :")
        submodules.display_table(ls_in_dis, [""] + hd_in)

        # Menampilkan daftar tipe pemasukan
        print("Daftar Tipe pengeluaran :")
        submodules.display_table(ls_ex_dis, [""] + hd_ex)

        # Tampilan menu subprogram
        print("")
        print("""Pilih menu:
1. Buat tipe baru
2. Hapus tipe
3. Edit batas presentase pengeluaran
4. Keluar dari subprogram""")

        # Input pilihan menu dari user
        pilih_menu = submodules.input_of_int_options("Silakan pilih menu 1,2,3,4", [1,2,3,4])

```

Bagian akses tipe pendapatan/pengeluaran, menampilkan daftar tipe pemasukan/pengeluaran, tampilan menu subprogram, input pilihan menu pada fungsi edit\_tipe.

```

# Pilihan buat tipe baru
if pilih_menu == 1:
    kode = submodules.input_of_int_options("Pilih buat tipe pemasukan (1) atau tipe pengeluaran (0) ", [0,1])
    buat_tipe(kode)

# Pilihan menghapus tipe yg ada
elif pilih_menu == 2:
    pilih_label = submodules.input_of_int_options(f"Input nomor tipe (1-{byk_tipe1+byk_tipe2}) yang ingin dihapus ", list(range(1,byk_tipe1+byk_tipe2+1)))

    # Menghapus tipe pemasukan/pemasukan berdasarkan var pilih_label
    if pilih_label <= byk_tipe1:
        tipe_dihapus = ls_in[pilih_label-1][0]
        dari = "Pemasukan"
    else:
        tipe_dihapus = ls_ex[pilih_label-byk_tipe1-1][0]
        dari = "Pengeluaran"

    # Konfirmasi hapus tipe
    print(f"Konfirmasi hapus tipe:
Tipe {submodules.ch_color_style(tipe_dihapus, "blue")} dari {submodules.ch_color_style(dari, "blue")}")

    konfir_hapus = submodules.input_of_yatidak("Yakin ingin menghapus tipe? (y/t) ")
    # Jika konfir diterima, hapus tipe dan tulis ulang file tipe
    if konfir_hapus == "y":
        if pilih_label <= byk_tipe1:
            ls_in.pop(pilih_label-1)
            submodules.open_write_all_csv("tipe_pemasukan.csv", ls_in, hd_in)
        else:
            ls_ex.pop(pilih_label-byk_tipe1-1)
            submodules.open_write_all_csv("tipe_pengeluaran.csv", ls_ex, hd_ex)

    # Kembali ke menu subprogram

```

Bagian pilih buat/hapus tipe, konfirmasi hapus tipe pada fungsi edit\_tipe.

```

# Pilihan menu edit batas presentase pengeluaran
elif pilih_menu == 3:
    # Pilih pengeluaran yg ada
    pilih_label = submodules.input_of_int_options(f"Input nomor tipe pengeluaran ({byk_tipe1+1}-{byk_tipe1+byk_tipe2}) yang ingin diganti ")

    # Tanyakan presentase baru yg diharapkan
    while True:
        try:
            prc_baru = float(input("Masukkan batas presentase baru (Contoh Input: 90, 75, 55.5) \n"))
            assert prc_baru >= 0 and prc_baru <= 100, "Presentase salah!"
            break
        except AssertionError as er:
            print(er)
        except ValueError:
            print("Input tidak Valid!")

    # Konfirmasi input perubahan presentase
    dis_prc_baru = f"{prc_baru}%"
    print(f"Konfirmasi ubah batas presentase:
Nama tipe pengeluaran {submodules.ch_color_style(ls_ex[pilih_label-byk_tipe1-1][0], "blue")}
Batas presentase baru {submodules.ch_color_style(dis_prc_baru, "blue")}")

    konfir_ganti = submodules.input_of_yatidak("Yakin ingin menyimpan perubahan? (y/t) ")
    # apabila konfir diterima, ubah presentase dari tipe terpilih dan tulis ulang file tipe
    if konfir_ganti == "y":
        ls_ex[pilih_label-byk_tipe1-1][1] = prc_baru
        submodules.open_write_all_csv("tipe_pengeluaran.csv", ls_ex, hd_ex)

# Pilihan untuk keluar dari subprogram
elif pilih_menu == 4:
    konfir_keluar = submodules.input_of_yatidak("Yakin ingin keluar dari subprogram? (y/t) ")
    if konfir_keluar == "y":
        return # Keluar dari subprogram

```

Bagian pilih pengeluaran, menanyakan presentase, konfirmasi presentase, dan keluar dari subprogram pada fungsi edit\_tipe.

```

import submodules
from datetime import date, timedelta
from os import remove, getcwd
from os.path import join

```

Bagian import pada subprogram invoice.

```

def invoice():
    # Identitas Subprogram
    print("\n"+"INVOICE".center(50,"=")+"\n")

    # Persiapan tabel invoice
    hd_iv = ["", "Tanggal", "Dompet", "Tipe", submodules.ch_color_style("Pemasukan", "green"), submodules.ch_color_style("Pengeluaran", "red")]
    ls_iv = []

    # Buka file sejarah transaksi
    _, ls_tr = submodules.open_read_csv("sejarah_transaksi.csv")

    # Restriksi 30 hari yg lalu
    hari_ini = date.today()
    hari_30_belakang = (hari_ini - timedelta(days=30)).strftime("%Y/%m/%d")
    ls_tr_new = []
    for ele in ls_tr:
        if ele[0] < hari_30_belakang:
            break
        ls_tr_new.append(ele)

    # Cek transaksi 30 hari ada
    if len(ls_tr_new) == 0:
        print("Kamu belum membuat transaksi 30 hari ini!")
        print("Tidak bisa melakukan pembuatan invoice")
        return

```

Bagian identitas subprogram, persiapan tabel invoice, buka file sejarah transaksi, restriksi 30 hari, cek transaksi 30 hari, pada fungsi invoice.

```

# Pembuatan baris invoice pertabel
for id, ele in enumerate(ls_tr_new):
    if ele[1] == "1":
        ls_iv.append([ id+1, ele[0], ele[3], ele[2], submodules.ch_color_style(f"Rp{int(ele[4]):>10,}", "green"), None ])
    else:
        ls_iv.append([ id+1, ele[0], ele[3], ele[2], None, submodules.ch_color_style(f"Rp{int(ele[4]):>10,}", "red") ])

# Menampilkan tabel invoice
print("Daftar transaksi per 30 hari terakhir : ")
submodules.display_table(ls_iv, hd_iv)

# Tanya apakah akan membuat invoice pdf
buat_pdf = submodules.input_of_yatidak("Apakah mau membuat pdf invoice? (y/t) ")
if buat_pdf == "y":
    from pdf_1 import buat_pdf_1 # Pemanggilan pembuat pdf tabel invoice
    from pdf_2_3 import buat_pdf_2_3 # Pemanggilan pembuatan pdf grafik keuangan
    from pdf_4_5 import buat_pdf_4_5 # Pemanggilan pembuatan pdf chart lingkaran

    buat_pdf_1()
    buat_pdf_2_3()
    buat_pdf_4_5()

    from PyPDF2 import PdfMerger # Mulai penggabungan pdf

    # Daftar file pdf
    pdf_files = ['pdf_1.pdf', 'pdf_2.pdf', 'pdf_3.pdf', 'pdf_4.pdf', 'pdf_5.pdf']

    # Mulai penggabungan pdf
    merger = PdfMerger()

    # Penggabungan pdf pada merger
    for file in pdf_files:
        merger.append(file)

```

Bagian pembuatan baris invoice, menampilkan tabel invoice, membuat pdf invoice, menggabungkan pdf, penggabungan pdf pada merger pada fungsi invoice.

```
# Deklarasi nama pdf gabungan
output_pdf = 'pdf_merge.pdf'

# Tulis merger ke nama pdf gabungan
merger.write(output_pdf)

# Tutup merger
merger.close()

# Hapus pdf lama
for ele in pdf_files + ["pdf_2.png", "pdf_3.png"]:
    remove(join(getcwd(),ele))

# Buka file pdf (Ada kemungkinan tidak berhasil)
import webbrowser
pdf_file = 'pdf_merge.pdf'
webbrowser.open_new_tab(pdf_file)
```

Bagian nama pdf gabungan dan menulis pdf gabungan.

### 3.10 Analisis keuangan

```
from . import submodules as sdl
from datetime import date, timedelta

def analisis_keuangan():
    # Identitas subprogram
    print("\n"+" ANALISIS KEUANGAN ".center(50,"=")+"\n")

    # Buka dan baca file sejarah_transaksi dan dompet
    hd_dp, ls_dp = sdl.open_read_csv("dompet.csv")
    _, ls_tr = sdl.open_read_csv("sejarah_transaksi.csv")

    # Restriksi 30 hari yg lalu
    hari_ini = date.today()
    hari_30_belakang = (hari_ini - timedelta(days=30)).strftime("%Y/%m/%d")
    ls_tr_new = [] # list untuk menampung list transaksi 30 hari ke belakang
    for ele in ls_tr:
        if ele[0] < hari_30_belakang:
            break
        ls_tr_new.append(ele)

    # Cek transaksi 30 hari apakah ada
    if len(ls_tr_new) == 0:
        print("Kamu belum membuat transaksi 30 hari ini!")
        print("Tidak bisa melakukan analisis keuangan")
        return # Mengakhiri program karena tidak ada transaksi
```

Import submodules, import kelas date dan timedelta dari modul datetime, mencetak identitas subprogram, membuka file dan dan baca file sejarah\_transaksi dan dompet, restriksi 30 hari yang lalu, mengecek transaksi rentang waktu 30 hari.



```

# Pertambahan/pengurangan dompet
hd_dp.append("Perubahan") # kolom untuk perubahan dompet
for i in range(len(ls_tr_new)):
    # Akses setiap kolom pada list transaksi
    _, kode, _, dompet, nominal = ls_tr_new[i]
    nominal = int(nominal)
    tr_index = -1 # Index -1
    for j in range(len(ls_dp)):
        if dompet == ls_dp[j][0]: # cari index untuk tr_index
            tr_index = j
            break
    try: # Tambahkan nominal pada transaksi apabila index 2 ls_dp ada
        ls_dp[tr_index][2] = ls_dp[tr_index][2] + nominal if kode == "1" else ls_dp[tr_index][2] - nominal
    except IndexError: # Jika ls_dp index 2 tidak ada
        (ls_dp[tr_index]).append(nominal) if kode == "1" else (ls_dp[tr_index]).append(-nominal)

# Ganti warna untuk perubahan dompet
for ele in ls_dp:
    try:
        ele[1] = f"Rp{int(ele[1]):>10,}"
        color2 = "green" if ele[2] >= 0 else "red"
        ele[2] = sdl.ch_color_style(f"Rp{ele[2]:>10,}", color2)
    except IndexError: # Apabila index 1/2 tidak ada
        pass

```

Bagian penambahan/pengurangan dompet dan mengganti warna untuk perubahan dompet.

```

# Jumlahkan pendapatan berdasarkan tipe dan jumlahkan pengeluaran berdasarkan tipe
hd_in = ["Pemasukan", "Subtotal"] # Header distribusi pendapatan
ls_in = []
hd_ex = ["Pengeluaran", "SubTotal", "% Thd pemasukan"] # Header distribusi pengeluaran
ls_ex = []

for i in range(len(ls_tr_new)):
    code = ls_tr_new[i][1]
    tipe = ls_tr_new[i][2]
    amount = int(ls_tr_new[i][4])

    if tipe == "transfer_akun":
        continue # Mengabaikan tipe transfer akun

    if code == "1": # Jumlahkan pendapatan berdasarkan tipe pada blok ini
        in_index = -1 # index -1
        for p in range(len(ls_in)):
            if ls_in[p][0] == tipe: # cari index untuk in_index
                in_index = p
                break
        if in_index != -1: # Tambahkan nominal jika index bukan -1
            ls_in[in_index][1] += amount
        else: # Append tipe, nominal jika index -1
            ls_in.append([tipe, amount])

    else: # Penjumlahan pengeluaran berdasarkan tipe pada blok ini
        ex_index = -1 # index -1
        for j in range(len(ls_ex)):
            if ls_ex[j][0] == tipe:
                ex_index = j
                break
        if ex_index != -1: # cari index untuk ex_index
            ls_ex[ex_index][1] += amount
        else: # Append tipe, nominal jika index -1
            ls_ex.append([tipe, amount])

```



Bagian menjumlahkan pendapatan berdasarkan tipe dan menjumlahkan pengeluaran berdasarkan tipe.

```
# Penambahan total semua tipe pendapatan
total_in = sum([ ele[1] for ele in ls_in ])
ls_in.append(["Total Seluruh", total_in])

# Penambahan total semua tipe pengeluaran
total_ex = sum([ ele[1] for ele in ls_ex ])
ls_ex.append(["Total Seluruh", total_ex])

for ele1 in ls_in:
    ele1[1] = f"Rp{int(ele1[1]):>10,}"

# Presentase pengeluaran pada blok ini
# Ambil batas presentase pada file tipe_pengeluaran
_, ls_tpex = sdl.open_read_csv("tipe_pengeluaran.csv")
notes = [] # Catatan peringatan di sini

for j in range(len(ls_ex)-1): # Mengecualikan total seluruh ex
    amount = int(ls_ex[j][1])
    tipe = ls_ex[j][0]

    # Default batas presentase pengeluaran per tipe 25 % oleh seluruh pendapatan
    # Ubah presentase tipe apabila ada
    prc = 25 # Default presentase 25%
    for k in range(len(ls_tpex)): # Akses apabila ada presentase kustom
        if ls_tpex[k][0] == tipe:
            prc = float(ls_tpex[k][1])

    # Hitung presentase pengeluaran per tipe thd pendapatan
    by_in = round(amount/total_in*100,2)
    ls_ex[j].append(f"{by_in}%")

    ls_ex[j][1] = f"Rp{amount:>10,}"

    if tipe == "bayar utang":
        continue

    if by_in > prc:
        for i in range(len(ls_ex[j])):
            ls_ex[j][i] = sdl.ch_color_style(ls_ex[j][i], "red")
        notes.append(f"Pengeluaran tipe {ls_ex[j][0]} berada di atas {prc}% pendapatan 30 hari ke belakang")
```

Bagian penambahan total semua tipe pendapatan, penambahan total semua tipe pengeluaran, mengambil batas persentase pada file tipe\_pengeluaran, default batas persentase pengeluaran per tipe 25% oleh seluruh pendapatan, mengubah persentase tipe bila ada, menghitung persentase pengeluaran per tipe terhadap pendapatan.

```

# Untuk total pengeluaran, batas presentase 70% thd seluruh pendapatan
all_amount = int(ls_ex[-1][1])
prc_all = 70 # Presentase seluruh pengeluaran dgn batas 70% thd pendapatan

by_in = round(all_amount/total_in*100,2)
ls_ex[-1].append(f"{by_in}%")

ls_ex[-1][1] = f"Rp{all_amount:>10,}"
if by_in > prc_all:
    for i in range(len(ls_ex[-1])):
        ls_ex[-1][i] = sdl.ch_color_style(ls_ex[-1][i],"red")
    notes.append(f"Pengeluaran total berada di atas {prc_all}% pendapatan 30 hari ke belakang")

# Tampilkan update dompet
print("Analisis Dompet 30 hari ke belakang :")
sdl.display_table(ls_dp, hd_dp)
print()

# Tampilkan pendapatan berdasarkan tipe
print("Analisis Pendapatan berdasarkan tipe 30 hari ke belakang :")
sdl.display_table(ls_in, hd_in)
print()

# Tampilkan pengeluaran berdasarkan tipe
print("Analisis Pengeluaran berdasarkan tipe 30 hari ke belakang :")
sdl.display_table(ls_ex, hd_ex)
print()

```

Bagian membatasi seluruh pengeluaran dengan batas 70% terhadap pendapatan, menampilkan update dompet, menampilkan pendapatan berdasarkan tipe, menampilkan pengeluaran berdasarkan tipe.

```

# Tampilkan peringatan
print("Catatan:")
if len(notes) == 0:
    print("Selamat, tidak ada catatan untuk pengeluaranmu!")
    print("Tetap gunakan uang dengan bijak")
else:
    for ele in notes:
        print(ele) # Tampilkan setiap peringatan yang ada

if __name__ == "__main__":
    analisis_keuangan()

```

Bagian menampilkan peringatan.

### 3.11 Pemasukan\_pengeluaran

```

import submodules as sdl
from pilih_dompet import pilih_dompet, tulis_dompet
from edit_tipe import pilih_tipe
from datetime import date

```

Bagian import submodules pada subprogram pemasukan\_pengeluaran.

```
def rekap_pemasukan_pengeluaran(code:int, tipe:str, dompet:str, nominal:int):
    # Akses tanggal pembuatan transaksi (hari ini)
    tanggal = (date.today()).strftime("%Y/%m/%d")

    # Persiapan list ditulis ke csv
    toadd = [[ tanggal, code, tipe, dompet, nominal ]]

    # Penulisan ke csv pada index value pertama
    sdl.open_append2first_csv("sejarah_transaksi.csv", toadd)

    # Penulisan/overwriting dompet
    tulis_dompot(dompot, code, nominal)
```

(variable) toadd: list[list]

Bagian fungsi rekap\_pemasukan\_pengeluaran dalam subprogram pemasukan\_pengeluaran.

```
def pemasukan_pengeluaran():
    while True: # Pengulangan selama belum keluar subprogram
        # Identitas Subprogram
        print("\n"+"PEMASUKAN PENGELUARAN".center(50,"=")+"\n")

        #pilih dompet
        pilih, nominalawal = pilih_dompot()

        #input 0/1
        inputpp = sdl.input_of_int_options("Pilih pemasukan (1) atau pengeluaran (0) ?", [0,1])

        #masukan nominal pemasukan/pengeluaran
        if inputpp == 1 :
            nominal = sdl.input_money("Masukkan nominal pemasukan :")
            clr = "sky"
            tr = "pemasukan"
        elif inputpp == 0 :
            nominal = sdl.input_money_w_params("Masukkan nominal pengeluaran :", 0, nominalawal)
            if nominal == None:
                print(sdl.ch_color_style("Uang tidak cukup, dikembalikan ke menu utama","yellow"))
                break # Ketika nominal pengeluaran gagal didapatkan
            clr = "red"
            tr = "pengeluaran"

        #memilih tipe transaksi
        pilih_tipe = pilih_tipe(inputpp)

        # Konfirmasi input
        dis_nom = f"Rp{nominal:,}"

        print(f'''Konfirmasi pembuatan transaksi:
Transaksi {sdl.ch_color_style(tr,clr)}
Tipe {sdl.ch_color_style(pilih_tipe,clr)}
Nominal {sdl.ch_color_style(dis_nom,clr)} pada dompet {sdl.ch_color_style(pilih,"sky")}
''')
```

Bagian fungsi pemasukan\_pengeluaran pada subprogram pemasukan\_pengeluaran.

```
# finalisasi pembuatan transaksi
final = sdl.input_of_yatidak("Finalisasi pembuatan transaksi ?(y/t)")
if final == "y" : # Jika konfir input diterima
    rekap_pemasukan_pengeluaran(inputpp, pilih tipe, pilih, nominal)
    break
# Apakah mau melanjutkan subprogram
elif final == "t" : # Jika konfir input tidak diterima
    transbaru = sdl.input_of_yatidak("buat transaksi baru? (input t untuk keluar dari subprogram) (y/t)")
    if transbaru == "t":
        break
```

Bagian finalisasi pada subprogram pemasukan\_pengeluaran.

### 3.12 Menampilkan\_dompét\_utang

```
import submodules as sdl
```

Bagian import submodules pada subprogram menampilkan\_dompét\_utang.

```
def menampilkan_dompét_utang():
    # Akses file domept dan file utang
    hd_dompét , ls_dompét = sdl.open_read_csv("dompet.csv")
    hd_utang , ls_utang = sdl.open_read_csv("utang.csv")

    # Penyesuaian konten nominal dengan mata uang rupiah
    ls_dompét = [ [ele[0], f"Rp{int(ele[1]):>10,}"] for ele in ls_dompét ]
    ls_utang = [ [ele[0], f"Rp{int(ele[1]):>10,}"] for ele in ls_utang ]

    # Tampilkan dompét
    if len(ls_dompét) >= 1:
        print("Daftar Dompét Pengguna :")
        sdl.display_table(ls_dompét, hd_dompét)
        print()
    else:
        print("\nKamu belum memiliki dompét\n")

    # Tampilkan utang
    if len(ls_utang) >= 1:
        print("Daftar Utang Pengguna :")
        sdl.display_table(ls_utang, hd_utang)
        print()
    else:
        print("\nSelamat, Kamu tidak memiliki utang!\n")
```

Bagian fungsi menampilkan dompét utang pada subprogram  
menampilkan\_dompét\_utang.

## BAB IV

### HASIL RUNNING PYTHON

```

===== SELAMAT DATANG DI PROGRAM =====
===== BUDGETING DAN PENCATATAN =====
===== KEUANGAN PRIBADI =====

Kamu belum memiliki dompet

Selamat, Kamu tidak memiliki utang!

==== =====
..                               Menu
==== =====
1  Pemasukan / Pengeluaran
2      Transfer Akun
3      Buat Utang
4      Bayar Utang
5      Edit Tipe
6      Analisis Keuangan
7      Invoice
0      Keluar
==== =====

Input 1-7 untuk pilih menu atau 0 untuk keluar

```

Tampilan menu utama saat memulai program

```

=====PEMASUKAN PENGELUARAN=====

Anda belum punya dompet!
Diarahkan ke bagian buat dompet
Masukkan nama dompet :
tunai
Konfirmasi pembuatan dompet:
Nama dompet = tunai
Apakah mau menyimpan dompet? (y/t)
y
Daftar dompet dapat dipilih :
==== =====
..      nama      nominal
==== =====
1  tunai  Rp      0
0  buat
==== =====
Input 1 untuk pilih dompet atau 0 untuk buat dompet
1
Pilih pemasukan (1) atau pengeluaran (0) ?
1
Masukkan nominal pemasukan :
100000
Daftar Tipe dapat dipilih:
==== =====
..      nama tipe
==== =====
1      gaji
2      uang saku
3      bonus/hadiah
4      parttime
5      dan lain-lain
0      buat
==== =====
Input 1-5 untuk pilih tipe Pemasukan atau 0 untuk buat tipe
2

```

Tampilan subprogram pemasukan/pengeluaran dengan input nama dompet "tunai",  
pemasukan, tipe "uang saku", nominal 100000

```

=====PEMASUKAN PENGELUARAN=====

Daftar dompet dapat dipilih :
=====
..      nama      nominal
=====
1      tunai      Rp 100,000
0      buat
=====
Input 1 untuk pilih dompet atau 0 untuk buat dompet
1
Pilih pemasukan (1) atau pengeluaran (0) ?
0
Masukkan nominal pengeluaran :
80000
Daftar Tipe dapat dipilih:
=====
..      nama tipe
=====
1      makan
2      jajan
3      transportasi
4      rekreasi
5      hadiah
6      komunikasi
7      dan lain-lain
0      buat
=====
Input 1-7 untuk pilih tipe Pengeluaran atau 0 untuk buat tipe
1
Konfirmasi pembuatan transaksi:
Transaksi pengeluaran
Tipe makan
Nominal Rp80,000 pada dompet tunai

Finalisasi pembuatan transaksi ?(y/t)
|

```

Tampilan subprogram pemasukan/pengeluaran dengan input nama dompet "tunai",  
pengeluaran, tipe "makan", nominal 80000

```

=====PEMASUKAN PENGELUARAN=====

Daftar dompet dapat dipilih :
=====
..      nama      nominal
=====
1      tunai      Rp 20,000
0      buat
=====
Input 1 untuk pilih dompet atau 0 untuk buat dompet
0
Masukkan nama dompet :
emoney
Konfirmasi pembuatan dompet:
Nama dompet = emoney
Apakah mau menyimpan dompet? (y/t)
y
Daftar dompet dapat dipilih :
=====
..      nama      nominal
=====
1      tunai      Rp 20,000
2      emoney      Rp 0
0      buat
=====
Input 1-2 untuk pilih dompet atau 0 untuk buat dompet
2
Pilih pemasukan (1) atau pengeluaran (0) ?
1
Masukkan nominal pemasukan :
50000
Daftar Tipe dapat dipilih:
=====
..      nama tipe
=====
1      gaji
2      uang saku
3      bonus/hadiah
4      parttime

```

Tampilan subprogram pemasukan/pengeluaran dengan input nama dompet "emoney",  
pemasukan, tipe "bonus/hadiah", nominal 50000

```
===== SELAMAT DATANG DI PROGRAM =====
===== BUDGETING DAN PENCATATAN =====
===== KEUANGAN PRIBADI =====

Daftar Dompot Pengguna :
=====
nama      nominal
=====
tunai    Rp    20,000
emoney   Rp    50,000
=====

Selamat, Kamu tidak memiliki utang!

====  =====
..      Menu
=====
1  Pemasukan / Pengeluaran
2      Transfer Akun
3      Buat Utang
4      Bayar Utang
5      Edit Tipe
6      Analisis Keuangan
7      Invoice
0      Keluar
=====

Input 1-7 untuk pilih menu atau 0 untuk keluar
```

Tampilan menu utama setelah pemasukan/pengeluaran dan inputnya

```
===== TRANSFER AKUN =====

====  =====
..      nama      nominal
=====
1  tunai    Rp    20,000
2  emoney   Rp    50,000
=====

Input 1-2 untuk memilih dompet pemberi transfer
2
Masukkan jumlah uang ditransfer
20000
Input 1-2 untuk memilih dompet penerima transfer
1
Masukkan biaya admin, input 0 jika tidak ada
0
Konfirmasi transfer akun :
Transfer sebanyak Rp20,000
Dari emoney
ke tunai
Yakin melakukan transfer akun? (y/t)
|
```

Tampilan subprogram transfer akun dengan input dompet pemberi "emoney", dompet penerima "tunai", nominal transfer 20000 tanpa biaya admin

```
===== SELAMAT DATANG DI PROGRAM =====
===== BUDGETING DAN PENCATATAN =====
===== KEUANGAN PRIBADI =====

Daftar Dompot Pengguna :
=====
nama      nominal
=====
tunai    Rp    40,000
emoney   Rp    30,000
=====

Selamat, Kamu tidak memiliki utang!

=====
..          Menu
=====
1 Pemasukan / Pengeluaran
2      Transfer Akun
3      Buat Utang
4      Bayar Utang
5      Edit Tipe
6      Analisis Keuangan
7      Invoice
0      Keluar
=====

Input 1-7 untuk pilih menu atau 0 untuk keluar
```

Tampilan menu utama setelah transfer akun dan inputnya

```
===== BUAT UTANG =====

Daftar dompet dapat dipilih :
=====
..      nama      nominal
=====
1  tunai  Rp    40,000
2  emoney Rp    30,000
0      buat
=====

Input 1-2 untuk pilih dompet atau 0 untuk buat dompet
2
Masukkan nama utang:
utang toko
Masukkan nominal utang:
40000
Konfirmasi pembuatan transaksi:
Transaksi pemasukan
Tipe utang
Nominal Rp40,000 pada dompet emoney

Apakah Anda ingin finalisasi pembuatan utang? (y/t)
y\
```

Tampilan subprogram buat utang dengan input nama utang "utang toko", dompet "emoney", nominal utang 40000



```
===== SELAMAT DATANG DI PROGRAM =====
===== BUDGETING DAN PENCATATAN =====
===== KEUANGAN PRIBADI =====

Daftar Dompot Pengguna :
=====
nama          nominal
=====
tunai  Rp      40,000
emoney Rp      70,000
=====

Daftar Utang Pengguna :
=====
nama          nominal
=====
utang toko  Rp      40,000
=====

====  =====
..      Menu
====  =====
1  Pemasukan / Pengeluaran
2      Transfer Akun
3      Buat Utang
4      Bayar Utang
5      Edit Tipe
6      Analisis Keuangan
7      Invoice
0      Keluar
====  =====

Input 1-7 untuk pilih menu atau 0 untuk keluar
|
```

Tampilan menu utama setelah buat utang dan inputnya

```
=====BAYAR UTANG=====

====  =====
..      nama          nominal
====  =====
1  utang toko  Rp      40,000
0      keluar
====  =====

Input 1 untuk utang dibayar atau 0 untuk keluar
1
Daftar dompet dapat dipilih :
====  =====
..      nama          nominal
====  =====
1  tunai  Rp      40,000
2  emoney Rp      70,000
0      buat
====  =====

Input 1-2 untuk pilih dompet atau 0 untuk buat dompet
1
Masukkan nominal bayar utang?
15000
Konfirmasi bayar utang:
nama utang = utang toko
nominal utang = Rp40,000
dibayar = Rp15,000 dengan dompet tunai
Apakah mau membayar utang? (y/t)
y|
```

Tampilan subprogram bayar utang dengan input nama utang dibayar "utang toko",  
dompet pembayar "tunai", nominal bayar utang 15000

```
===== SELAMAT DATANG DI PROGRAM =====
===== BUDGETING DAN PENCATATAN =====
===== KEUANGAN PRIBADI =====

Daftar Dompot Pengguna :
=====
nama      nominal
=====
tunai    Rp    25,000
emoney   Rp    70,000
=====

Daftar Utang Pengguna :
=====
nama      nominal
=====
utang toko Rp    25,000
=====

====  =====
..      Menu
====  =====
1  Pemasukan / Pengeluaran
2      Transfer Akun
3      Buat Utang
4      Bayar Utang
5      Edit Tipe
6      Analisis Keuangan
7      Invoice
0      Keluar
====  =====

Input 1-7 untuk pilih menu atau 0 untuk keluar
```

Tampilan menu utama setelah bayar utang dan inputnya

```
=====EDIT TIPE=====

Daftar Tipe Pemasukan :
=====
..      nama tipe
=====
1      gaji
2      uang saku
3      bonus/hadiah
4      parttime
5      dan lain-lain
=====

Daftar Tipe pengeluaran :
=====
..      nama tipe      presentase
=====
6      makan          25
7      jajan           25
8      transportasi    25
9      rekreasi        25
10     hadiah          25
11     komunikasi      25
12     dan lain-lain   25
=====

Pilih menu subprogram:
1. Buat tipe baru
2. Hapus tipe
3. Edit batas presentase pengeluaran
4. Keluar dari subprogram
Silakan pilih menu 1,2,3,4
```

Tampilan subprogram edit tipe

```

===== ANALISIS KEUANGAN =====

Analisis Dompot 30 hari ke belakang :
=====
nama      nominal      Perubahan
=====
tunai    Rp    40,000    Rp    40,000
emoney   Rp    70,000    Rp    70,000
=====

Analisis Pendapatan berdasarkan tipe 30 hari ke belakang :
=====
Pemasukan      Subtotal
=====
      utang    Rp    40,000
      bonus/hadiah Rp    50,000
      uang saku Rp   100,000
Total Seluruh  Rp   190,000
=====

Analisis Pengeluaran berdasarkan tipe 30 hari ke belakang :
=====
Pengeluaran      SubTotal      % Thd pemasukan
=====
      makan    Rp    80,000      42.11%
Total Seluruh  Rp    80,000      42.11%
=====

Catatan:
Pengeluaran tipe makan berada di atas 25.0% pendapatan 30 hari ke belakang
Tekan Enter untuk kembali ke menu utama

```

Tampilan subprogram analisis keuangan

```

===== INVOICE =====

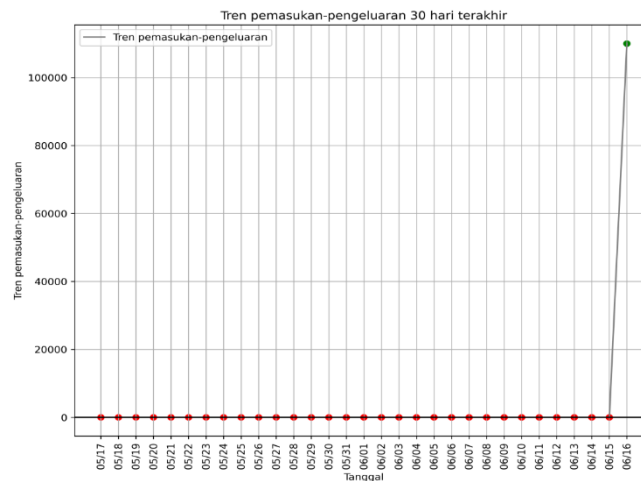
Daftar transaksi per 30 hari terakhir :
=====
..      Tanggal      Dompot      Tipe      Pemasukan      Pengeluaran
=====
1  2023/06/16      emoney      utang      Rp    40,000
2  2023/06/16      tunai      transfer_akun Rp    20,000
3  2023/06/16      emoney      transfer_akun Rp           Rp    20,000
4  2023/06/16      emoney      bonus/hadiah Rp    50,000
5  2023/06/16      tunai      makan      Rp           Rp    80,000
6  2023/06/16      tunai      uang saku   Rp   100,000
=====
Apakah mau membuat pdf invoice? (y/t)

```

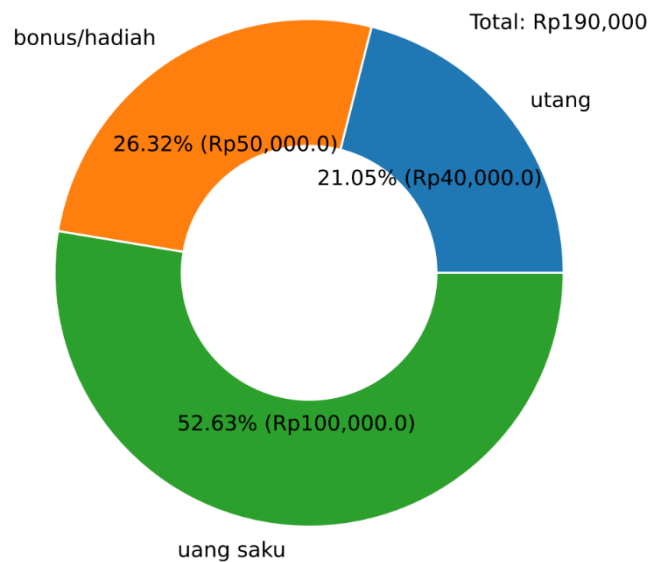
Tampilan subprogram invoice

Tabel sejarah transaksi 30 hari terakhir

	Tanggal	Domet	Tipe	pemasukan	pengeluaran
1	2023/06/16	emoney	utang	Rp 40,000	
2	2023/06/16	tunai	transfer_akun	Rp 20,000	
3	2023/06/16	emoney	transfer_akun		Rp 20,000
4	2023/06/16	emoney	bonus/hadiah	Rp 50,000	
5	2023/06/16	tunai	makan		Rp 80,000
6	2023/06/16	tunai	uang saku	Rp 100,000	



Distribusi pemasukan dalam 30 hari terakhir



Conoth hasil pdf invoice berupa laporan keuangan