CRUD PHYTON

CRUD PHYTON
CRUD PHYTON

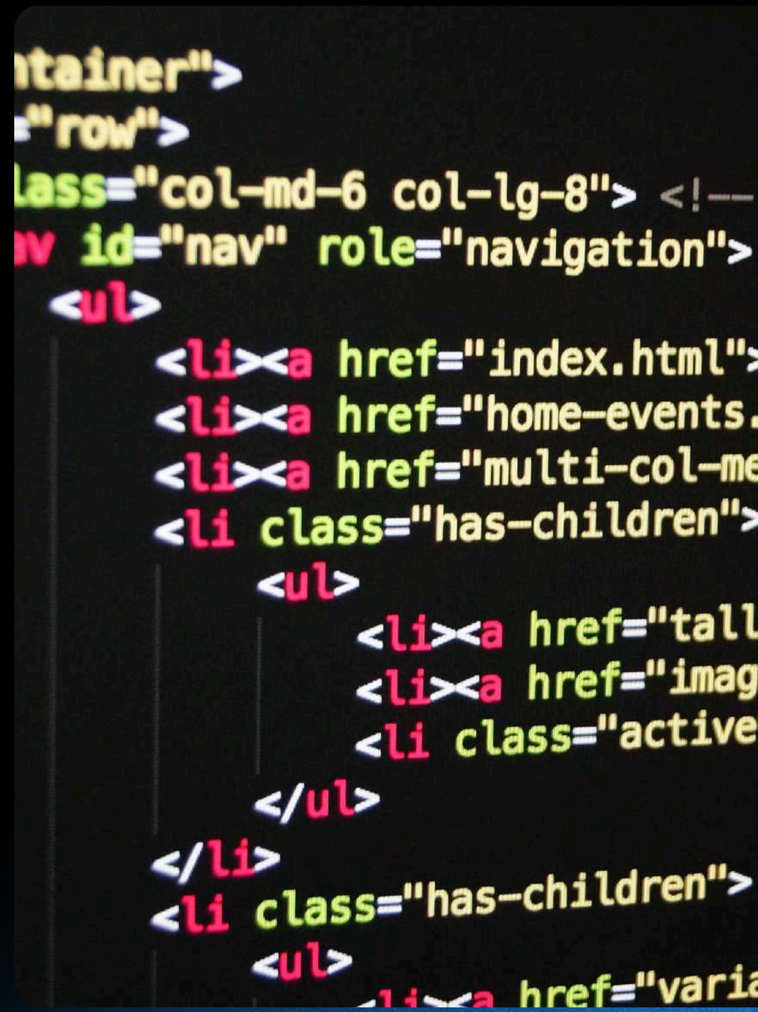
Giovanni Gunawan Wangidjaja

Apa Sebenarnya Program Ini dan Apa Tujuannya?

Konsep Utama: Program ini adalah simulasi digital dari struk belanja yang memungkinkan pengguna untuk berinteraksi dengan daftar belanjaan melalui antarmuka baris perintah (Command-Line Interface).

Tujuan Fungsional: Tujuannya adalah menyediakan fungsi CRUD, yaitu empat pilar operasi data dasar:

- Create (Membuat): Menambahkan barang baru ke dalam daftar belanja.
- Read (Membaca): Menampilkan seluruh daftar belanja atau mencari barang spesifik.
- Update (Memperbarui): Mengedit detail barang yang sudah ada (misal: mengubah jumlah).
- Delete (Menghapus): Menghilangkan barang dari daftar belanja.



Inisialisasi & Data Global

```python
from colorama import Fore, Style, init
init(autoreset=True)  # Inisialisasi colorama agar warna reset otomatis setelah print

# Data awal contoh struk pembelanjaan, berisi list dictionary tiap barang
struk = [
    {"id": "A001", "tipe": "Saniteri", "barang": "Sabun Batang", "jumlah": 12, "harga": 10000},
    {"id": "A002", "tipe": "Saniteri", "barang": "Sabun Cair", "jumlah": 2, "harga": 15000},
    {"id": "B001", "tipe": "Minuman", "barang": "Sirup", "jumlah": 1, "harga": 30000},
]

# Kamus tipe barang untuk pilihan input tipe barang berdasarkan nomor
tipe_barang_dict = {
    1: "Saniteri",
    2: "Minuman",
    3: "Makanan",
    4: "Elektronik",
    5: "Pakaian"
}
```

from colorama import ...: Mengimpor modul eksternal untuk memberikan warna pada teks terminal.

- Fore digunakan untuk warna teks (misal: Fore.GREEN).
- init(autoreset=True) adalah konfigurasi penting yang secara otomatis mengembalikan gaya teks ke default setelah setiap perintah print, sehingga kita tidak perlu me-resetnya secara manual.

struk = [...]: Ini adalah sumber data utama (Single Source of Truth) dari program.

- Struktur Data: Dipilih sebagai list dari dictionary. Ini adalah pilihan yang sangat baik karena:
  - list secara alami merepresentasikan urutan item dalam struk.
  - dictionary memungkinkan kita menyimpan data terstruktur dengan pasangan kunci-nilai (seperti 'id', 'harga') untuk setiap barang, membuatnya sangat mudah dibaca dan diakses.

tipe_barang_dict = {...}: Ini adalah sebuah kamus (dictionary) pemetaan.

- Tujuan: Untuk menyederhanakan input pengguna. Pengguna cukup memasukkan angka 1 daripada mengetik "Saniteri", yang mengurangi risiko kesalahan ketik dan menjaga konsistensi data.

```python
# Fungsi untuk memformat angka menjadi format ribuan dengan titik (contoh: 10.000)
def format_ribuan(nilai):
    return f"{nilai:,}".replace(",", ".")

# Fungsi untuk menghitung panjang garis pemisah tabel berdasarkan lebar data struk
def hitung_panjang_garis():
    max_id = max((len(item["id"]) for item in struk), default=6)
    max_tipe = max((len(item["tipe"]) for item in struk), default=10)
    max_barang = max((len(item["barang"]) for item in struk), default=14)
    # Total panjang = lebar tiap kolom + margin tetap
    total_panjang = max_id + max_tipe + max_barang + 8 + 8 + 12 + 15
    return total_panjang + 5  # margin tambahan

# Fungsi menampilkan garis horizontal berwarna cyan
def garis(panjang=None):
    if panjang is None:
        panjang = hitung_panjang_garis()
    print(Fore.CYAN + '-' * panjang)

# Fungsi menampilkan header judul dengan garis atas dan bawah
def tampilkan_header_judul(judul):
    panjang = hitung_panjang_garis()
    garis(panjang)
    print(Fore.YELLOW + judul.center(panjang))
    garis(panjang)
```

Fungsi Bantuan (Helpers)

Fungsi-fungsi ini digunakan untuk membersihkan kode utama agar tidak terjadi pengulangan code.

format_ribuan(nilai): Mengubah angka seperti 10000 menjadi string 10.000 agar mudah dibaca.

hitung_panjang_garis(): Menghitung lebar tabel secara dinamis berdasarkan panjang data terpanjang. Ini membuat tabel tetap rapi meskipun data berubah.

garis() & tampilkan_header_judul(judul): Fungsi utilitas untuk mencetak garis pemisah dan header. Menggunakan fungsi ini membuat kode utama lebih bersih dan mudah dibaca

```python
# Fungsi menambah barang baru ke struk dengan validasi input
def tambah_barang():
    while True:
        panjang = hitung_panjang_garis()
        garis(panjang)
        id_baru = input(Fore.GREEN + "Masukkan ID Barang : " + Style.RESET_ALL).strip()
        # Validasi panjang ID
        if len(id_baru) < 4 or len(id_baru) > 6:
            print(Fore.RED + "ID Barang harus antara 4 sampai 6 karakter.")
            continue
        # Cek ID sudah ada atau belum
        if any(item["id"].lower() == id_baru.lower() for item in struk):...
        break

    # Menampilkan pilihan tipe barang dan input tipe baru
    print("Pilih Tipe Barang:")
    for nomor, tipe in tipe_barang_dict.items():...

    while True:...

    # Input nama barang dan validasi panjang
    nama_baru = input(Fore.GREEN + "Masukkan Nama Barang : " + Style.RESET_ALL).strip().title()
    if len(nama_baru) < 1 or len(nama_baru) > 50:...

    # Input jumlah barang dengan validasi angka dan batasan
    while True:...

    # Input harga barang dengan validasi angka dan batasan
    while True:...

    garis(panjang)
    yakin = input(Fore.GREEN + "Apakah Data Sudah Benar? (Y/N) : " + Style.RESET_ALL).strip().lower(
    if yakin == 'y':...
    else:
        print(Fore.RED + "Penambahan barang dibatalkan.")
```

Fungsi Inti (CREATE)

Fungsi ini adalah contoh bagus dari Defensive Programming, di mana program mengantisipasi input yang salah dari pengguna.

Loop Validasi while True: Pola ini digunakan berulang kali untuk terus meminta input kepada pengguna sampai input yang diberikan memenuhi kriteria yang ditetapkan (break). Jika tidak, continue akan mengulang loop.

Tiga Lapisan Validasi Utama:

- Validasi Format (ID): Memeriksa panjang string.
- Validasi Keunikan (ID): Menggunakan generator expression (item["id"]... for item in struk) di dalam fungsi any() untuk memeriksa secara efisien apakah ID sudah ada. any() berhenti sobald menemukan kecocokan pertama.

- Validasi Tipe & Rentang (Jumlah & Harga):
  - Blok try-except ValueError adalah cara standar Python untuk menangani input yang seharusnya angka tetapi diberi teks (misalnya, pengguna mengetik "lima" bukan 5).
  - Pemeriksaan if jumlah_baru < 1 or jumlah_baru > 99 adalah validasi rentang (business logic).

Mekanisme Penambahan: Jika semua validasi lolos dan pengguna mengonfirmasi, sebuah dictionary baru dibuat dan ditambahkan ke akhir list struk menggunakan metode .append().

Fungsi Inti (READ)

```python
# Fungsi menampilkan data struk dalam bentuk tabel ke terminal
def tampilkan_struk(data):
    panjang = hitung_panjang_garis()
    garis(panjang)
    header = f" {'ID':<6}| {'Tipe':<10}| {'Barang':<14}| {'Jumlah':<8}| {'Harga':<8}| {'Total':<12}"
    print(Fore.YELLOW + header)
    garis(panjang)
    for item in data:
        total = item["jumlah"] * item["harga"]  # Hitung total harga per item
        print(f" {item['id'].upper():<6}| {item['tipe']:<10}| {item['barang']:<14}| "
              f"{item['jumlah']:<8}| {format_ribuan(item['harga']):<8}| {format_ribuan(total):<12}")
    garis(panjang)

# Fungsi mencari barang berdasarkan ID (case insensitive)
def cari_barang_by_id(id_cari):
    hasil = [item for item in struk if item["id"].lower() == id_cari.lower()]
    return hasil
```

tampilkan_struk(data):

- Fleksibilitas: Didesain untuk menerima data (sebuah list) sebagai argumen. Ini membuatnya sangat bisa digunakan kembali; bisa untuk menampilkan seluruh struk atau hanya hasil pencarian.
- Kalkulasi "On-the-fly": Nilai total tidak disimpan di dalam data, melainkan dihitung saat akan ditampilkan (item["jumlah"] * item["harga"]). Ini menghemat memori dan memastikan total selalu akurat.
- Formatting Teks: Menggunakan f-string dengan spesifikasi format (:<6) untuk memastikan setiap kolom memiliki lebar yang rata (rata kiri), menciptakan tampilan tabel yang rapi.

cari_barang_by_id(id_cari):

- List Comprehension: Menggunakan sintaks list comprehension yang ringkas dan efisien. Baris [item for item in struk if ...] adalah cara Pythonic untuk membangun list baru berdasarkan kondisi tertentu.
- Case-Insensitive Search: Kondisi item["id"].lower() == id_cari.lower() membandingkan ID dalam format huruf kecil. Ini adalah praktik UX yang baik karena pengguna tidak perlu khawatir tentang kapitalisasi (misalnya, a001 akan menemukan A001).

Fungsi Inti (UPDATE)

```python
# Fungsi update data barang berdasarkan ID
def update_barang():
    id_edit = input(Fore.GREEN + "Masukkan ID Barang : " + Style.RESET_ALL).strip()
    barang_edit = cari_barang_by_id(id_edit)
    if not barang_edit:
        print(Fore.RED + "ID Barang Tidak Ditemukan.")
        return
    barang = barang_edit[0]

    tampilkan_struk([barang])  # Tampilkan data barang yang akan diedit

    yakin_update = input(Fore.GREEN + "Ketik Y jika ingin update atau N Jika Ingin Cancel Update (Y/N) : " + Style.RESET_ALL).strip().lower()
    if yakin_update != 'y':
        print(Fore.RED + "Update dibatalkan.")
        return

    kolom_edit = input(Fore.GREEN + "Masukkan Kolom yang Ingin di Edit\n(Tipe/Barang/Jumlah/Harga): " + Style.RESET_ALL).strip().lower()

    # Edit sesuai kolom yang dipilih
    if kolom_edit == "tipe":···
    elif kolom_edit == "barang":···
    elif kolom_edit == "jumlah":···
    elif kolom_edit == "harga":···
    else:···

    yakin_simpan = input(Fore.GREEN + "Apakah Anda Ingin Update Data (Y/N): " + Style.RESET_ALL).strip().lower()
    if yakin_simpan == 'y':
        print(Fore.GREEN + "Update Data Berhasil")
        tampilkan_struk(struk)
    else:
        print(Fore.RED + "Update dibatalkan.")
```

Langkah 1: Pencarian: Langkah pertama selalu menemukan data yang benar. Fungsi ini secara cerdas menggunakan kembali cari_barang_by_id() yang sudah kita buat.

Konsep Kunci: Referensi vs Salinan:

- Baris barang = barang_edit[0] sangat penting. Di Python, saat Anda mengambil objek yang bisa diubah (mutable) seperti dictionary dari sebuah list, barang tidak menjadi salinan. Ia menjadi sebuah referensi atau "penunjuk" ke dictionary asli yang ada di dalam list struk.
- Implikasi: Karena itu, ketika kita mengubah nilai barang["jumlah"] = jumlah_baru, kita secara langsung mengubah data di dalam list struk utama.
-

Logika Pengkondisian: if/elif/else digunakan sebagai router untuk menentukan bagian mana dari dictionary yang akan diperbarui berdasarkan input pengguna (kolom_edit).

```python
# Fungsi menghapus barang berdasarkan ID
def hapus_barang():
    id_hapus = input(Fore.GREEN + "Masukkan ID Barang yang ingin dihapus: " + Style.RESET_ALL).strip()
    barang_hapus = cari_barang_by_id(id_hapus)
    if not barang_hapus:
        print(Fore.RED + "ID Barang Tidak Ditemukan.")
        return
    struk.remove(barang_hapus[0])  # Hapus barang dari list
    print(Fore.GREEN + "Barang Berhasil Dihapus")
    tampilkan_struk(struk)
```

Fungsi Inti (DELETE)

Proses yang Mirip dengan Update: proses ini diawali dengan mencari item yang akan dihapus menggunakan cari_barang_by_id.

Error Handling: Perintah if not barang_hapus: adalah guard clause yang penting. Ia memeriksa apakah hasil pencarian kosong. Jika ya, ia mencetak pesan kesalahan dan menggunakan return untuk menghentikan eksekusi fungsi lebih lanjut.

Metode .remove():

- Aksi penghapusan inti dilakukan oleh struk.remove(barang_hapus[0]).
- Metode .remove() pada list akan mencari elemen pertama yang identik dengan argumen yang diberikan (dalam kasus ini, dictionary barang yang ditemukan) dan menghapusnya dari list secara in-place (langsung memodifikasi list asli).

Umpan Balik Pengguna: Setelah menghapus, program memberikan konfirmasi ("Barang Berhasil Dihapus") dan memanggil tampilkan_struk(struk) untuk menunjukkan kepada pengguna keadaan terbaru dari data mereka.

Alur Program Utama

def main(): Mengemas logika utama program ke dalam satu fungsi adalah praktik yang baik. Ini membuat kode lebih terstruktur dan dapat diuji.

while True: (Loop Utama): Ini adalah event loop dari aplikasi kita. Ia bertanggung jawab untuk:

- Terus-menerus menampilkan menu utama.
- Menunggu input pengguna.
- Mengarahkan program ke fungsi yang sesuai.
- Loop ini hanya berhenti jika pengguna memilih '5', yang memicu perintah break.

Struktur Menu Bertingkat (Nested Loop): Perhatikan ada while True di dalam if pilihan == '1'. Ini menciptakan sub-menu. Pengguna bisa melakukan beberapa aksi di dalam "Menu Lihat" (misal: lihat semua, lalu cari ID) sebelum akhirnya memilih '3' untuk break dari loop sub-menu dan kembali ke loop menu utama.

if __name__ == "__main__":: Ini adalah konstruksi standar di Python.

- Fungsi: Ia memastikan bahwa fungsi main() hanya akan dieksekusi ketika file program.py dijalankan secara langsung (python program.py).
- Manfaat: Jika di masa depan kita ingin mengimpor fungsi dari file ini ke file lain (misalnya, mengimpor format_ribuan), kode di dalam main() tidak akan berjalan secara otomatis. Ini membuat kode kita bisa digunakan kembali sebagai modul.

```python
# Fungsi utama program yang menjalankan loop menu utama
def main():
    while True:
        pilihan = tampilkan_menu_utama()
        if pilihan == '1':
            while True:
                pilihan_lihat = tampilkan_menu_lihat()
                if pilihan_lihat == '1':
                    tampilkan_struk(struk)  # Tampilkan semua barang
                elif pilihan_lihat == '2':
                    id_cari = input(Fore.GREEN + "Masukkan ID Barang yang ingin dicari: " + Style.RESET_ALL).strip()
                    hasil = cari_barang_by_id(id_cari)
                    if hasil:
                        tampilkan_struk(hasil)
                    else:
                        print(Fore.RED + "Data tidak ditemukan.")
                elif pilihan_lihat == '3':
                    break  # Kembali ke menu utama
                else:
                    print(Fore.RED + "Pilihan tidak valid.")
        elif pilihan == '2':…
        elif pilihan == '3':…
        elif pilihan == '4':…
        elif pilihan == '5':…
        else:
            print(Fore.RED + "Pilihan tidak valid, coba lagi.")

if __name__ == "__main__":
    main()

```

DEMONSTRASI

THANK YOU