

Sistem Pencatatan Data Pemesanan Kereta

Menggunakan Bahasa Pemrograman Python

Anggota Kelompok:

Muhamad Ferdinal Raihan Abdullah - 12220586

Muhammad Rifky Maulana - 12220704

Ahmad Fariz - 12220131

Muhammad Syamsul Falah - 12221014

Anindya Imtiyaz - 12221461



Project Pencatatan Data Pemesanan Kereta

Tujuan utama dari pembuatan project ini adalah membuat sebuah sistem yang digunakan untuk mempermudah proses pencatatan data dalam pemesanan kereta

Program ini termasuk coding yang sederhana dengan metode pemrograman yang berorientasikan pada objek (OOP), menggunakan beberapa modul python seperti pandas dan datetime, menggunakan fungsi Perulangan dan Percabangan. Adapun beberapa pilar-pilar OOP yang kami terapkan dalam project ini, yaitu Inheritance (Pewarisan), dan Encapsulation (Enkapsulasi).

Tampilan Ketika Program Dijalankan

*Tampilan Awal

Tarif Kereta Jakarta - Yogyakarta			
	Bogowonto (B)	Senja Utama (S)	Fajar Utama (F)
Kelas			
Exclusive	Rp. 185,000,-	Rp. 156,000,-	Rp. 110,000,-
Business	Rp. 285,000,-	Rp. 235,000,-	Rp. 190,000,-
Economic	Rp. 275,000,-	Rp. 225,000,-	Rp. 185,000,-

Form Penginputan Data

Tanggal Input : 15 December 2022

Jumlah Data : 2

*Tampilan Ketika Penginputan Data

Form Penginputan Data	
Tanggal Input :	15 December 2022
Jumlah Data :	2
Data Ke - 1	
Nama Pemesan :	Rifky
Pilih Kelas [1/2/3] :	1
Pilih Jenis Kereta [B/S/F] :	F
Jumlah Penumpang :	15
Data Ke - 2	
Nama Pemesan :	Ferdinal
Pilih Kelas [1/2/3] :	3
Pilih Jenis Kereta [B/S/F] :	B
Jumlah Penumpang :	25

Tampilan Ketika Program Dijalankan

*Tampilan Akhir setelah penginputan data selesai

Data Pemesanan Kereta						
Tanggal : 15 December 2022						
	Nama Pemesan	Kelas	Jenis Kereta	Tarif Per Seat	Jumlah Penumpang	Total Harga
1	Rifky	Exclusive	Fajar Utama	Rp. 110,000,-	15	Rp. 1,650,000,-
2	Ferdinal	Economic	Bogowonto	Rp. 275,000,-	25	Rp. 6,875,000,-

Total Pendapatan : Rp. 8,525,000

Tahapan Pembuatan Program

Pada proses pembuatan project ini terdapat 3 proses atau tahapan, diantaranya:

- Proses Perancangan
- Proses Development (Pengembangan Program)
- Proses Debugging (Meminimalisir Error atau Bug)

Proses Perancangan

Pada proses perancangan program ini, pertama-tama kami menentukan bahasa pemrograman yang akan digunakan yaitu bahasa pemrograman Python.

Lalu, kami menentukan kelas dan jenis kereta terlebih dahulu. Kelas kereta dibagi menjadi 3 yaitu Economic, Business, dan Exclusive. Sedangkan jenis kereta dibedakan menjadi 3 yaitu Bogowonto (B), Senja Utama (S), dan Fajar Utama (F).



Proses Development

Proses ini adalah inti dari pembuatan project program, karena tujuan makalah ini dibuat adalah untuk memberikan penjelasan bagaimana kami membuat program ini, dan hasil program nya seperti apa.

Dalam proses ini kami membuat sebuah program dari referensi data yang sudah dijelaskan di Proses Perancangan.



Class Referensi

Class Referensi berisi referensi data-data dan method yang berkaitan dengan data-data yang diinputkan

```
● ● ●

class Referensi:
    # Struktur Data (Constructor) dari class Referensi
    def __init__(self):
        self.data_kelas = ('Exclusive', 'Business', 'Economic')
        self.data_b = (185000, 285000, 275000)
        self.data_s = (156000, 235000, 225000)
        self.data_f = (110000, 190000, 185000)

    # Fungsi untuk menampilkan Data Referensi
    def tampilan_referensi(self):
        referensi = {
            "Kelas" : self.data_kelas,
            "Bogowonto (B)" : self.data_b,
            "Senja Utama (S)" : self.data_s,
            "Fajar Utama (F)" : self.data_f
        }
        tabel_referensi = pd.DataFrame(referensi).set_index("Kelas")

        # Mengubah Data Harga Referensi ke format mata uang rupiah
        tabel_referensi[['Bogowonto (B)', 'Senja Utama (S)', 'Fajar Utama (F)']] =
        tabel_referensi[['Bogowonto (B)', 'Senja Utama (S)', 'Fajar Utama (F)']].apply(
            lambda series: series.apply(lambda value: "Rp. {:.2f},-".format(value)))
        self.atur_padding_tabel(['Bogowonto (B)', 'Senja Utama (S)', 'Fajar Utama (F)'],
        tabel_referensi)

        # Menampilkan Data Referensi beserta Garis Pembatas
        print(self.garis())
        print("Tarif Kereta Jakarta - Yogyakarta".center(75))
        print(self.garis())
        print(tabel_referensi)
        print(self.garis())
```

```
● ● ●

# Fungsi untuk mengatur padding pada tabel
def atur_padding_tabel(self, kolom, tabel, padding = 20):
    for key in kolom:
        tabel[key] = tabel[key].str.pad(padding, side='left')

# Fungsi untuk mengambil data kelas
def ambil_data_kelas(self, index):
    return self.data_kelas[index]

# Fungsi untuk mengambil data harga dari jenis kereta "Bogowonto (B)"
def ambil_data_b(self, index):
    return self.data_b[index]

# Fungsi untuk mengambil data harga dari jenis kereta "Senja Utama (S)"
def ambil_data_s(self, index):
    return self.data_s[index]

# Fungsi untuk mengambil data harga dari jenis kereta "Fajar Utama (F)"
def ambil_data_f(self, index):
    return self.data_f[index]

# Fungsi untuk membuat sebuah garis atau pembatas
def garis(self, jumlah_garis = 75):
    return "="*jumlah_garis
```

Class Data

Class Data berisi data-data yang diinputkan oleh user/pengguna yang nantinya akan dikalkulasikan berdasarkan data yang diberikan oleh Parent-nya yaitu (Class Referensi)

```
class Data(Referensi):

    # Attribut dari class Data
    data_nama = []
    data_kelas_pesanan = []
    data_jenis_kereta = []
    data_tarif = []
    data_penumpang = []
    data_jumlah_harga = []
    data_total_pendapatan = 0

    # Struktur Data (Constructor) dari class Referensi
    def __init__(self):
        # Mengambil struktur data (Constructor) dari Parent (Class Referensi)
        super().__init__()

    # Fungsi untuk menyimpan data yang diinputkan oleh user ke dalam Attribut dari class Data, agar
    nantinya bisa diakses oleh method lain
    def set_data(self, nama, kelas_pesanan, jenis_kereta, tarif, penumpang, jumlah_harga):
        self.data_nama.append(nama)
        self.data_kelas_pesanan.append(kelas_pesanan)
        self.data_jenis_kereta.append(jenis_kereta)
        self.data_tarif.append(tarif)
        self.data_penumpang.append(penumpang)
        self.data_jumlah_harga.append(jumlah_harga)

    # Fungsi untuk menyimpan data Total Pendapatan yang berisi penjumlahan dari seluruh data jumlah
    harga yang diinputka
    def set_total_pendapatan(self):
        self.data_total_pendapatan = sum(self.data_jumlah_harga)
```

```
# Fungsi untuk mengambil data dari attribut class Data dan mengubahnya ke dalam bentuk sebuah
Dictionary
def get_data(self):
    return {
        "Nama Pemesan" : self.data_nama,
        "Kelas" : self.data_kelas_pesanan,
        "Jenis Kereta" : self.data_jenis_kereta,
        "Tarif Per Seat" : self.data_tarif,
        "Jumlah Penumpang" : self.data_penumpang,
        "Total Harga" : self.data_jumlah_harga,
    }

# Fungsi untuk menampilkan Data keseluruhan
def tampilkan_data(self, tanggal):
    # Menampilkan judul beserta garis pembatas
    print(self.garis(135))
    print("Data Pemesanan Kereta".center(135))
    print(f"\nTanggal : {tanggal}")
    print(self.garis(135))

    # Mengubah data yang berbentuk Dictionary ke dalam bentuk tabel menggunakan Pandas DataFrame
    data = self.get_data()
    tabel_data = pd.DataFrame(data)
    # Mengubah Data Harga Tarif Per Seat, dan Total Harga ke format mata uang rupiah
    tabel_data[['Tarif Per Seat', 'Total Harga']] = tabel_data[['Tarif Per Seat', 'Total
Harga']].apply(
        lambda series: series.apply(lambda value: "Rp. {:.},-{:.format(value))")
    )
    # Mengatur padding pada tabel
    self.atur_padding_tabel(['Nama Pemesan', 'Kelas', 'Jenis Kereta', 'Tarif Per Seat', 'Total
Harga'], tabel_data)
    tabel_data.index += 1

    # Menampilkan tabel Data beserta garis pembatasnya
    print(tabel_data)
    print(self.garis(135))

    # Mengambil data total pendapatan, lalu merubah nilainya ke format mata uang rupiah
    self.set_total_pendapatan()
    print("Total Pendapatan : Rp. {:.},-{:.format(self.data_total_pendapatan))
```

Membuat Object dari Class

Setelah membuat 2 Class sebelumnya, karena menggunakan metode OOP. Kami membuat object yang merepresentasikan Class tersebut.



```
# Membuat object dan menampilkan Data Referensi dari Class
Referensi = Referensi()
referensi.tampilkan_referensi()

# Membuat object dari Class Data
data = Data()
```

Fungsi Validasi

Untuk memaksimalkan program yang kami buat, dibuatlah fungsi validasi yang bertujuan untuk meminimalisir error atau bug ketika user atau pengguna salah memasukkan data

*Validasi Input Kelas Kereta

```
● ● ●

# Fungsi untuk Validasi Input Kelas Kereta
def validasi_kelas_kereta(kelas):
    if((kelas > 0) & (kelas <= 3)):
        # Jika true, mengembalikan data kelas dari Class Referensi sesuai index
        # yang diinputkan
        return referensi.data_kelas[kelas-1]
    else:
        # Jika false, mengembalikan pesan error
        raise ValueError('Kelas yang anda masukkan salah, tolong masukkan Kelas
[1,2,3]')
```

*Validasi Input Jenis Kereta

```
● ● ●

# Fungsi untuk validasi input jenis kereta
def validasi_jenis_kereta(jenis, kelas):
    # Mengubah nilai jenis kereta yang diinputkan menjadi uppercase, untuk
    # menangani input yang tidak ter-CapsLock
    jenis = jenis.upper()

    # Jika true pada 3 Kondisi yang ditentukan, maka mengembalikan sebuah
    # dictionary yang berisi kata kunci jenis dan tarif. Tarif diambil dari class
    # Referensi berdasarkan jenis yang diinputkan dan index yang inputkan pada input
    # kelas sebelumnya
    if jenis == "B":
        return {
            "jenis" : "Bogowonto",
            "tarif" : referensi.data_b[kelas-1]
        }

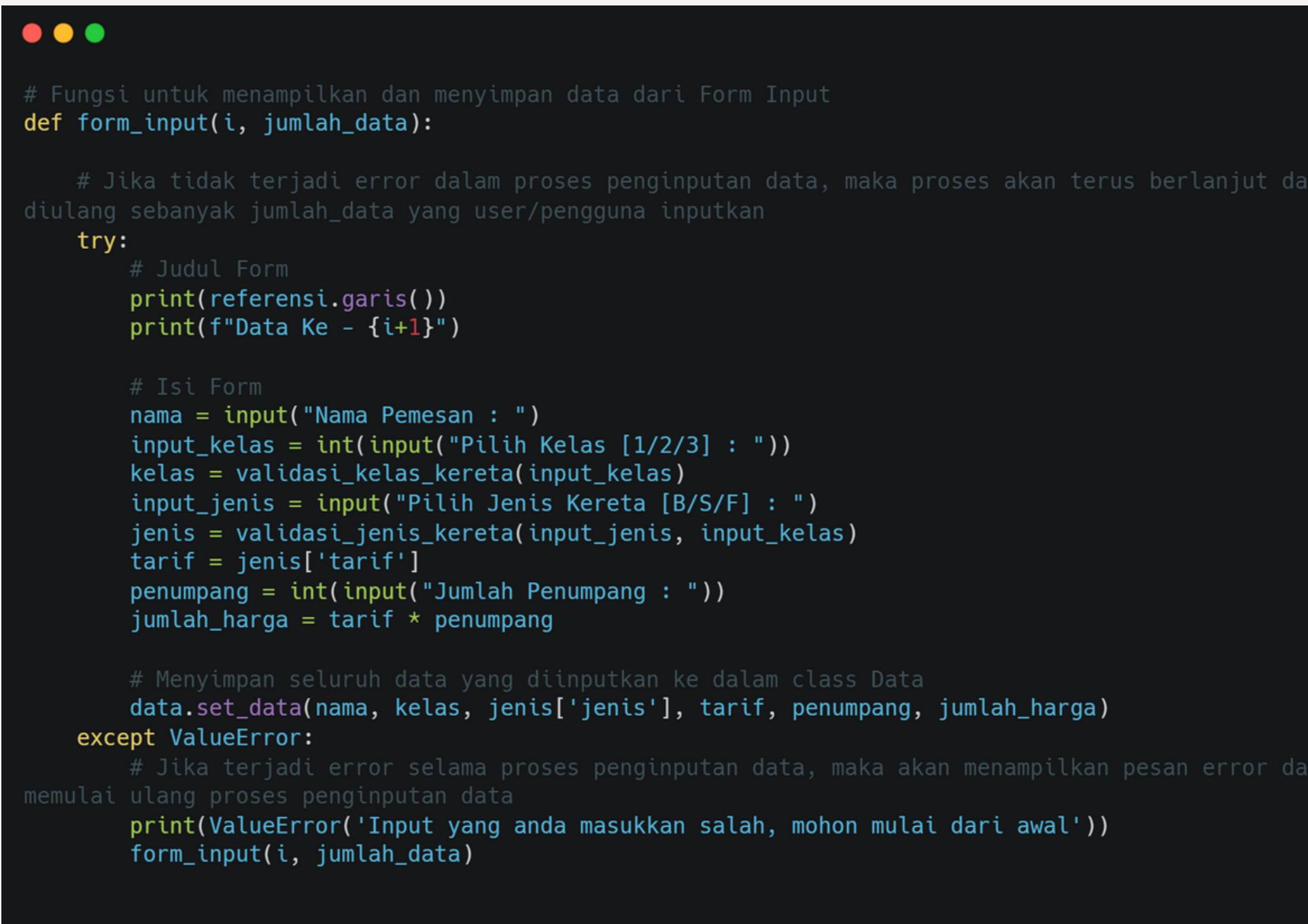
    elif jenis == "S":
        return {
            "jenis" : "Senja Utama",
            "tarif" : referensi.data_s[kelas-1]
        }

    elif jenis == "F":
        return {
            "jenis" : "Fajar Utama",
            "tarif" : referensi.data_f[kelas-1]
        }

    else:
        # Jika false pada 3 kondisi diatas, maka akan mengembalikan pesan error
        raise ValueError('Jenis Kereta yang anda masukkan salah, tolong masukkan
Jenis Kereta [B/S/F]')
```

Fungsi untuk Form Input

Karena di program yang kita buat ini setiap langkahnya terdapat sebuah validasi, maka kami membungkus form input ke dalam sebuah fungsi. Agar nantinya ketika terjadi error form input ini dapat dipanggil kembali untuk mengulang proses penginputan data.



```
# Fungsi untuk menampilkan dan menyimpan data dari Form Input
def form_input(i, jumlah_data):

    # Jika tidak terjadi error dalam proses penginputan data, maka proses akan terus berlanjut dan
    # diulang sebanyak jumlah_data yang user/pengguna inputkan
    try:
        # Judul Form
        print(referensi.garis())
        print(f"Data Ke - {i+1}")

        # Isi Form
        nama = input("Nama Pemesan : ")
        input_kelas = int(input("Pilih Kelas [1/2/3] : "))
        kelas = validasi_kelas_kereta(input_kelas)
        input_jenis = input("Pilih Jenis Kereta [B/S/F] : ")
        jenis = validasi_jenis_kereta(input_jenis, input_kelas)
        tarif = jenis['tarif']
        penumpang = int(input("Jumlah Penumpang : "))
        jumlah_harga = tarif * penumpang

        # Menyimpan seluruh data yang diinputkan ke dalam class Data
        data.set_data(nama, kelas, jenis['jenis'], tarif, penumpang, jumlah_harga)
    except ValueError:
        # Jika terjadi error selama proses penginputan data, maka akan menampilkan pesan error dan
        # memulai ulang proses penginputan data
        print(ValueError('Input yang anda masukkan salah, mohon mulai dari awal'))
        form_input(i, jumlah_data)
```

Fungsi untuk Menjalankan Program

Dalam menjalankan program ini, kami membuat sebuah fungsi. Karena ditahap awal terdapat validasi untuk mengecek apakah input jumlah data yang dimasukkan user atau pengguna sudah sesuai atau tidak sesuai. Jadi, ketika tidak sesuai maka fungsi ini akan digunakan kembali.

*Fungsi Menjalankan Program

```
●●●

def input_jumlah_data():
    # Input jumlah data yang akan diinputkan
    jumlah_data = int(input("Jumlah Data : "))
    if(jumlah_data > 0):
        return jumlah_data
    else:
        raise ValueError('Jumlah data minimal 1!')

def start_program():
    try:
        # Input jumlah data yang akan diinputkan
        jumlah_data = input_jumlah_data()

        # Melakukan perulangan sebanyak jumlah data yang diinputkan
        for i in range(jumlah_data):
            # Memanggil fungsi form_input dengan mengisi parameternya dengan index dari perulangan,
            dan jumlah data yang sudah diinputkan sebelumnya
            form_input(i, jumlah_data)

            # Menampilkan Seluruh Data yang sudah user/pengguna inputkan sebelumnya
            data.tampilkan_data(date.today().strftime('%d %B %Y'))
    except:
        # Jika terjadi kesalahan input di jumlah data, maka input akan diulang kembali
        print(ValueError("Jumlah data harus berupa angka dan minimal 1!"))
        start_program()
```

*Memanggil fungsi untuk menjalankan program

```
●●●

# Form Pengisian Data
print("Form Penginputan Data".center(75))

# Mengambil tanggal hari ini menggunakan Library Datetime, lalu
menampilkannya
tanggal_input = f"Tanggal Input : {date.today().strftime('%d %B %Y')}"
print(tanggal_input)

# Menjalankan Program
start_program()
```



Proses Debugging

Pada tahapan terakhir ini, kami melakukan testing program di setiap prosesnya. Mulai dari inputan pertama, hingga inputan terakhir. Apabila terjadi error pada saat proses penginputan data kami sesegera mungkin untuk memperbaiki program yang kami buat.

Contoh penerapannya adalah kami membuat fungsi validasi pada setiap langkah penginputan data

Penutup

Sekian presentasi dari kelompok kami. Jika ada yang ingin bertanya, lebih baik pendam saja, karena kami sudah menjelaskan semuanya. Terima kasih.