Laporan ALP Alpro
Aaron, Aryo, Deny

Coding	Fungsi	Penjelasan

`import os`
`import time`
`from datetime import datetime, timedelta`
`import threading`
`import tkinter as tk`
	Import Library	`os` digunakan untuk memberikan fungsionalitas untuk berinteraksi dengan sistem operasi yang sedang dijalankan.
		
		`time` digunakan untuk menyediakan fungsi-fungsi yang berkaitan dengan waktu dan penundaan.
		
		`datetime` digunakan untuk menyediakan kelas dan fungsi untuk bekerja dengan tanggal dan waktu. Sedangkan `timedelta` digunakan untuk melakukan operasi matematika pada waktu, seperti menambah atau mengurangkan selisih waktu dari objek `datetime`
		
		`threading` menyediakan fungsionalitas untuk mengimplementasikan threading (pemrograman dengan menggunakan utas atau thread) dalam program Python. Thread adalah unit kecil dari proses yang dapat berjalan secara bersamaan, memungkinkan eksekusi tugas-tugas secara paralel. Dalam program ini `import threading` digunakan untuk menjalankan hitung mundur sebelum notifikasi reminder muncul, serta menghitung mundur sebelum waktu booking habis.
		
		`import tkinter as tk` adalah cara untuk mengimpor modul `tkinter` di Python dan memberikan alias atau nama singkat `tk` ke modul tersebut. Modul `tkinter` adalah modul bawaan dalam Python yang menyediakan alat untuk membuat antarmuka pengguna grafis (GUI). Dengan mengimpor modul `tkinter` dengan alias `tk`, kita dapat menggunakan nama pendek `tk` saat memanggil fungsi dan kelas dari modul tersebut. Dalam program ini `tkinter` digunakan untuk membuat pop-up dengan GUI sederhana.

`daftar_pengguna = []` `def registrasi():`	Registrasi	Dilakukan deklarasi array kosong dengan nama `daftar_pengguna` untuk mengisi

print("Silakan registrasi.") username = input("Masukkan username baru: ").lower() password = input("Masukkan password baru: ") daftar_pengguna.append({'username' : username, 'password': password}) os.system('cls') print(loginpage) print("Registrasi berhasil.")		data-data pengguna. Dalam function registrasi ini akan diminta username dan password dari pengguna dengan menggunakan input. Setelah itu username menggunakan **.lower()** untuk mengubah inputan username menjadi huruf kecil agar memudahkan saat pengguna login. Setelah itu dilakukan append dari username dan password yang dimasukkan pengguna dengan melakukan **daftar_pengguna.append()**. Pembuatan dictionary dilakukan dengan menambahkan **{'username': username, 'password': password}** dalam array agar mudah dicek menggunakan for loop statement sekaligus username dan juga password pengguna diappend ke dalam array. Setelah diappend, layar dibersihkan menggunakan **os.system('cls')** dan UI loginpage diprint lalu ditambahkan notes "Registrasi berhasil."

def login():
print("Login")
username = input("Masukkan username: ").lower()
password = input("Masukkan password: ")
 	
# cek menggunakan for loop, jika if sesuai maka hasilnya kembali true, jika tidak maka hasil kembali false 	
for i in daftar_pengguna:
if i['username'] == username and i['password'] == password:
os.system('cls')
	Login	Dalam function login ini akan diminta username dan juga password yang sudah diappend ke dalam array **daftar_pengguna** pada tahap registrasi.
		
		Untuk mengecek data yang sesuai dalam **daftar_pengguna** dilakukan statement loop for dan conditional if yaitu **for i in daftar_pengguna:**
		if i['username'] == username and i['password'] == password:.
		
		for i in daftar_pengguna: digunakan untuk membuat variabel i menjadi isi dari array **daftar_pengguna** serta berfungsi sebagai loop. Lalu setelah itu dilakukan pengecekan dengan **if i['username'] == username and i['password'] == password:**, yaitu

print(loginpage) print(f"Login berhasil!\nSelamat datang {username}!") return True os.system('cls') print(loginpage) print("Username atau password salah. Login gagal atau akun belum terdaftar.") return False		jika variabel i yang berdictionary username di dalam **daftar_pengguna** sesuai dengan username inputan user (**if i['username'] == username**) dan password di dalam **daftar_pengguna** sesuai dengan password inputan user (**and i['password'] == password**) akan ditampilkan kalimat **Login berhasil! \nSelamat datang !** (\n disini digunakan untuk membuat baris baru) dan akan **return True**. Tetapi jika data yang dimasukkan tidak sesuai maka akan ditampilkan kalimat **Username atau password salah. Login gagal atau akun belum terdaftar.** dan akan **return False**. **return True** dan **return False** disini akan digunakan pada statement **if login():** yang dimana jika **if True** maka perintah selanjutnya akan dijalankan, **if False** perintah selanjutnya tidak akan dijalankan.

def buat_parkir(kapasitas):
slots = ['kosong'] * kapasitas
#Mengisi beberapa slot sebagai contoh
slots[0:5] = ['terisi']*(5)
slots[55:60] = ['terbooking']*(5)
slots[32:37] = ['terbooking']*(5)
slots[73:78] = ['terisi']*(5)
return slots
 	
#Menentukan kapasitas parkir
kapasitas_parkir = 100
#Memanggil fungsi untuk membuat slot parkir
slot_parkir =
	Buat Slot Parkir	Function **buat_parkir** dimulai dengan mendeklarasikan sebuah fungsi yang menerima satu parameter, yaitu **kapasitas**, yang mewakili jumlah total slot parkir.
		
		Di dalam fungsi, di buat list yang bernama **slots** yang berisi string **'kosong'** sebanyak ‘kapasitas’ yang sudah ditentukan sebelumnya yaitu **kapasitas_parkir = 100**.
		
		Selanjutnya, beberapa slot dipilih sebagai contoh untuk diisi dengan status **'terisi'** dan **'terbooking'**. Sebagai ilustrasi, **slots[0:5] = ['terisi']*(5)** digunakan untuk mengisi slot parkir dengan status ‘terisi’ untuk slot 0 hingga 4. **slots[55:60] = ['terbooking']*(5)** digunakan untuk mengisi slot parkir dengan status ‘terbooking’ untuk slot 55 hingga 59. **slots[32:37] = ['terbooking']*(5)** digunakan untuk

`buat_parkir(kapasitas_parkir)`		mengisi slot parkir dengan status ‘terbooking’ untuk slot 32 hingga 36. `slots[73:78] = ['terisi']*(5)` digunakan untuk mengisi slot parkir dengan status ‘terisi’ untuk slot 73 hingga 77. `return slots` ini mengembalikan list `slots` yang mencerminkan status akhir dari semua slot parkir setelah beberapa di antaranya diisi sebagai ‘terisi’ atau ‘terbooking’. Diluar fungsi ini, `kapasitas_parkir` ditetapkan sebagai 100, dan fungsi `buat_parkir` dipanggil dengan menggunakan kapasitas tersebut untuk membuat dan menginisialisasi status slot parkir. Hasilnya disimpan dalam variabel `slot_parkir`.

`def lihat_slot_kosong(slots):`
`for i, status in enumerate(slots):`
`if status == 'kosong':`
 	
`print(f"\x1b[37m{chr(65 + i // 10)}{i % 10 + 1}\x1b[0m", end=' ')`
`#Mengubah warna angka slot menjadi putih`
`elif status == 'booking':`
 	
`print(f"\x1b[33;4;1m{chr(65 + i // 10)}{i % 10 + 1}\x1b[0m", end=' ')`
`#Mengubah warna angka slot menjadi kuning`
`elif status == 'merah':`
	Lihat Slot Parkir	Function `lihat_slot_kosong` ini menerima satu parameter yaitu `slots`. Di dalam fungsi ini terdapat loop for yang menggunakan enumerate untuk mendapatkan indeks i dan nilai status dari setiap elemen dalam slots.
		
		Selanjutnya, pada bagian `if status == 'kosong':` , `elif status == 'booking':`, dst. Menggunakan penyataan kondisional untuk memeriksa status setiap slot dan mencetak informasi dengan warna tertentu sesuai dengan statusnya. Code ini menggunakan kode ANSI escape untuk mengubah warna teks di terminal.
		Jika status slot adalah 'kosong', maka

```
print(f"\x1b[31;4;1m{chr(65 + i // 10)}{i % 10 + 1}\x1b[0m", end=' ')
#Mengubah warna angka slot menjadi merah
            elif status == 'terbooking':

  print(f"\x1b[33m{chr(65 + i // 10)}{i % 10 + 1}\x1b[0m", end=' ')
#Mengubah warna angka slot menjadi kuning
        elif status == 'terisi':

  print(f"\x1b[31m{chr(65 + i // 10)}{i % 10 + 1}\x1b[0m", end=' ')
#Mengubah warna angka slot menjadi merah

        if (i + 1) % 10 == 0:
  #Membuat baris baru setelah 10 slot
            print()

pilihan = input("Masukkan pilihan (1/2/3/4/5/6): ")
        if pilihan == '1':

        os.system('cls')
                print('='*30)

  lihat_slot_kosong(slot_parkir)

  input(f"{'='*30}\nTekan Enter untuk Kembali")
```

outputnya akan dicetak dengan warna putih.
Jika status slot adalah 'booking', maka outputnya akan dicetak dengan warna kuning dan cetak tebal.
Jika status slot adalah 'merah', maka outputnya akan dicetak dengan warna merah dan cetak tebal.
Jika status slot adalah 'terbooking', maka outputnya akan dicetak dengan warna kuning.
Jika status slot adalah 'terisi', maka outputnya akan dicetak dengan warna merah.

`if (i + 1) % 10 == 0:`
`print()`
pernyataan ini digunakan untuk memeriksa apakah `(i + 1)` habis di bagi 10, dan jika True maka **`print()`** akan menambahkan baris baru ke output. Bagian ini berfungsi agar setelah mencetak 10 slot, maka akan membuat baris baru untuk membuat tata letak yang lebih rapi.

`pilihan = input("Masukkan pilihan (1/2/3/4/5/6): ")`
ini akan mengambil input dari pengguna yang diharapkan berupa angka (1, 2, 3, 4, 5, atau 6) yang kemudian akan disimpan dalam variabel pilihan.
`if pilihan == '1':`
ini akan memeriksa **`pilihan`** yang dimasukkan oleh pengguna adalah `'1'`. Jika kondisi benar maka perintah selanjutnya akan dijalankan.

Selanjutnya, digunakan `os.system('cls')` untuk memberihkan

		layar terminal `lihat_slot_kosong(slot_parkir)` digunakan untuk fungsi `lihat_slot_kosong` dengan parameter `slot_parkir` . Fungsi ini kemudian akan mencetak informasi tentang status slot parkir. `input(f"{'='*30}\nTekan Enter untuk Kembali")` ini digunakan untuk menampilkan pesan input di prompt dengan 30 karakter sama dengan dan menunggu sampai pengguna menekan Enter sebelum melanjutkan.

`def parkir(slots, slot, sudah_parkir, sudah_booking):`
`if 1 <= slot <= len(slots):`
`if not sudah_parkir:`
`if sudah_booking and slots[slot - 1] == 'booking':`
`confirmation = input(f"Apakah Anda ingin parkir di slot \033[34m{inp_slot}\033[0m yang telah Anda booking sebelumnya? (Y/N): ")`
`if confirmation.upper() == 'Y':`
`slots[slot - 1] = 'merah'  #Mengubah status slot menjadi 'merah'`
`print(f"Mobil berhasil diparkir di slot \033[34m{inp_slot}\033[0m")`
`sudah_parkir =`
	Parkir	`def parkir` Membuat fungsi parkir dengan empat parameter: slots (array status slot parkir), slot (nomor slot yang ingin diisi), sudah_parkir (boolean apakah pengguna sudah melakukan parkir sebelumnya), dan sudah_booking (boolean apakah pengguna sudah melakukan booking sebelumnya).
		
		`if 1 <= slot <= len(slots):` Memeriksa apakah nomor slot yang dimasukkan berada dalam rentang yang valid, yaitu antara 1 dan panjang array slots.
		
		`if not sudah_parkir:`
		Memeriksa apakah pengguna belum melakukan parkir sebelumnya.
		
		`if sudah_booking and slots[slot - 1] == 'booking':`
		Memeriksa apakah pengguna telah melakukan booking sebelumnya dan apakah status slot yang dipilih adalah 'booking'.

```
                True
                    #Append
                History

    waktu_sekarang=datetime.now()

      history.append(f"Parkiran
    \033[34m{inp_slot}\033[0m diisi
                pada
{waktu_sekarang.strftime("%Y-%m-%d
            %H:%M")}")
                    elif
     confirmation.upper() == 'N':
                    print("Parkir
          dibatalkan.")
                    time.sleep(2)
             elif not sudah_booking
 and slots[slot - 1] == 'kosong':
                slots[slot - 1] =
  'merah'  #Mengubah status slot
          menjadi 'merah'
                 print(f"Mobil
     berhasil diparkir di slot
    \033[34m{inp_slot}\033[0m")
                  sudah_parkir =
                True
                  #Append History

    waktu_sekarang=datetime.now()

      history.append(f"Parkiran
    \033[34m{inp_slot}\033[0m diisi
                pada
{waktu_sekarang.strftime("%Y-%m-%d
            %H:%M")}")
                    else:
                print("Slot sudah
```

```
confirmation = input(f"Apakah
Anda ingin parkir di slot
\033[34m{inp_slot}\033[0m yang
telah Anda booking sebelumnya?
(Y/N): ")
```

Meminta konfirmasi dari pengguna apakah mereka ingin parkir di slot yang telah mereka booking sebelumnya.

```
if confirmation.upper() == 'Y':
```

Jika pengguna mengkonfirmasi (memilih 'Y'), maka status slot tersebut diubah menjadi 'merah', dan pesan sukses dicetak. Selain itu, waktu parkir juga dicatat di history menggunakan `history.append()`.

```
elif confirmation.upper() ==
'N':
```

Jika pengguna memilih untuk tidak parkir (memilih 'N'), maka pesan pembatalan dicetak.

```
elif not sudah_booking and
slots[slot - 1] == 'kosong':
```

Jika pengguna belum melakukan booking sebelumnya dan status slot yang dipilih adalah 'kosong', maka status slot diubah menjadi 'merah', dan pesan sukses dicetak. Waktu parkir juga dicatat di history menggunakan `history.append()`.

```
else:
    print("Slot
sudah terisi.")
```

Jika kondisi di atas tidak terpenuhi, artinya slot sudah terisi, dan pesan "Slot sudah terisi" dicetak.

```
else:
print("Anda sudah
melakukan parkir sebelumnya.")
```

Jika pengguna sudah melakukan parkir

terisi.") else: print("Anda sudah melakukan parkir sebelumnya.") time.sleep(2) else: print("Slot parkir tidak valid.") time.sleep(2) return sudah_parkir		sebelumnya, pesan "Anda sudah melakukan parkir sebelumnya." dicetak. **else:** **print("Slot parkir tidak valid.")** Jika nomor slot tidak valid, pesan "Slot parkir tidak valid." dicetak. **return sudah_parkir** Mengembalikan nilai sudah_parkir setelah selesai melakukan operasi di atas.

pilihan = input("Masukkan pilihan (1/2/3/4/5/6): ")
elif pilihan == '2':
if not sudah_booking and not sudah_parkir: #cek sudah booking/parkir, kalau ada yang false maka kode ini tidak berjalan
os.system('cls')
print('='*30)
lihat_slot_kosong(slot_parkir)
print('='*30)
inp_slot = input("Masukkan nomor slot yang akan dibooking: ").upper()
slot = konversi_input_slot(inp_slot)
if 1 <= slot <= len(slot_parkir) and
	Booking / Cancel Booking	**pilihan = input("Masukkan pilihan (1/2/3/4/5/6): ")**
		ini akan mengambil input dari pengguna yang diharapkan berupa angka (1, 2, 3, 4, 5, atau 6) yang kemudian akan disimpan dalam variabel pilihan.
		elif pilihan == '2':
		Memeriksa apakah pilihan pengguna adalah '2'.
		if not sudah_booking and not sudah_parkir:
		Memeriksa apakah pengguna belum melakukan booking dan belum melakukan parkir sebelumnya.
		os.system('cls')
		Digunakan untuk embersihkan layar konsol.
		lihat_slot_kosong(slot_parkir)
		Memanggil fungsi lihat_slot_kosong untuk menampilkan slot parkir yang kosong.
		inp_slot = input("Masukkan nomor slot yang akan dibooking: ").upper()
		Meminta input dari pengguna berupa nomor slot yang akan dibooking dan mengonversinya

slot_parkir[slot - 1] == 'kosong': #cek apakah slot yang dimasukkan ada / valid / sesuai dengan rentang array yg sudah di buat , and cek status slot yang dipilih kosong atau tidak slot_parkir[slot - 1] = 'booking' #ubah status yang sebelumnya kosong jadi booking pada slot yang dipilih sudah_booking = True #ubah status variabel sudah booking menjadi true notibooking() # TIMER thread = threading.Timer(30 ,timer_booking_timeout, args=(slot, slot_parkir)) thread.start() threadreminder = threading.Timer(15 , notireminder) threadreminder.start() #append history waktu_sekarang = datetime.now() waktu_selesai = datetime.now() + timedelta(minutes=30)		ke huruf besar (upper case). slot = konversi_input_slot(inp_slot) Memanggil fungsi konversi_input_slot untuk mengubah input slot menjadi format yang sesuai. if 1 <= slot <= len(slot_parkir) and slot_parkir[slot - 1] == 'kosong': Memeriksa apakah nomor slot yang dimasukkan valid (sesuai rentang array) dan status slot tersebut adalah 'kosong'. slot_parkir[slot - 1] = 'booking' Mengubah status slot yang dipilih menjadi 'booking'. sudah_booking = True Mengubah variabel sudah_booking menjadi True karena pengguna telah melakukan booking. notibooking() Memanggil fungsi notibooking untuk memberikan notifikasi booking. thread = threading.Timer(30 ,timer_booking_timeout, args=(slot, slot_parkir)) Membuat objek thread untuk mengatur timer booking timeout selama 30 detik. thread.start() Memulai thread timer. threadreminder = threading.Timer(15 , notireminder) Membuat objek thread untuk mengatur timer

history.append(f"Parkiran \033[34m{inp_slot}\033[0m dibooking pada {waktu_sekarang.strftime("%Y-%m-%d %H:%M")} dan akan berakhir pada {waktu_selesai.strftime("%Y-%m-%d %H:%M")}")
else:
#jika slot yang di input tidak sesuai
 	
print("Slot yang Anda pilih tidak tersedia atau sudah terisi.")
 	
time.sleep(2)
elif sudah_booking and not sudah_parkir:
cancel = input(f"Apakah Anda ingin membatalkan booking untuk slot \033[31m{inp_slot}\033[0m? (Y/N): ")
if cancel.upper() =='Y':
if slot_parkir[slot - 1] == 'booking':
 	
slot_parkir[slot - 1] = 'kosong'
 	
sudah_booking = False
 	
print(f"Booking slot \033[31m{inp_slot}\033[0m berhasil dibatalkan.")
		reminder selama 15 detik.
		threadreminder.start()
		Memulai thread timer reminder.
		
		waktu_sekarang = datetime.now()
		Mendapatkan waktu saat ini.
		
		waktu_selesai = datetime.now() + timedelta(minutes=30)
		Mendapatkan waktu selesai booking (30 menit setelah waktu saat ini).
		
		Menambahkan informasi booking ke dalam history menggunakan **history.append()**.
		
		else:
		print("Slot yang Anda pilih tidak tersedia atau sudah terisi.")
		Jika nomor slot tidak sesuai, mencetak pesan bahwa slot yang dipilih tidak tersedia atau sudah terisi.
		
		elif sudah_booking and not sudah_parkir:
		Jika pengguna sudah melakukan booking dan belum parkir, memasuki blok ini.
		
		cancel = input(f"Apakah Anda ingin membatalkan booking untuk slot \033[31m{inp_slot}\033[0m? (Y/N): ")
		Meminta konfirmasi dari pengguna apakah mereka ingin membatalkan booking.
		
		if cancel.upper() =='Y':
		Jika pengguna mengkonfirmasi (memilih 'Y'), membatalkan booking dan mengubah status slot menjadi 'kosong'.
		
		elif cancel.upper() == 'N':
		JIka pengguna memilih N maka booking tidak jadi dibatalkan

#Append History waktu_sekarang=datetime.now() history.append(f"Booking slot \033[31m{inp_slot}\033[0m dibatalkan pada {waktu_sekarang.strftime("%Y-%m-%d %H:%M")}") elif cancel.upper() == 'N': print() else: #jika user sudah booking atau parkir. print("Anda sudah melakukan parkir atau booking sebelumnya.") time.sleep(2)		else: print("Anda sudah melakukan parkir atau booking sebelumnya.") Jika pengguna sudah melakukan parkir atau booking sebelumnya, mencetak pesan bahwa mereka sudah melakukan parkir atau booking.

def keluar_parkir(slots, sudah_parkir, sudah_booking):
if sudah_parkir:
confirmation = input(f"Apakah Anda ingin mengeluarkan mobil dari slot \033[31m{inp_slot}\033[0m? (Y/N): ")
if confirmation.upper() =='Y':
slots[slot-1] = 'kosong'
print(f"Mobil berhasil
	Keluar	def keluar_parkir
		Membuat fungsi keluar_parkir dengan tiga parameter: slots (array status slot parkir), sudah_parkir (boolean apakah mobil sudah parkir), dan sudah_booking (boolean apakah mobil sudah dibooking).
		
		if sudah_parkir:
		Memeriksa apakah mobil sudah parkir.

```
        keluar dari slot
    \033[31m{inp_slot}\033[0m")
                #Append history

    waktu_sekarang=datetime.now()
            history.append(f"Anda
        keluar dari parkiran
    \033[31m{inp_slot}\033[0m pada
{waktu_sekarang.strftime("%Y-%m-%d
            %H:%M")}")
             sudah_booking = False
              sudah_parkir = False
                 return slots,
    sudah_parkir, sudah_booking
          elif confirmation.upper()
               == 'N':
                print("Pengeluaran
        mobil dibatalkan.")
                 return slots,
    sudah_parkir, sudah_booking
                   else:
              print("Masukan tidak
      valid. Pengeluaran mobil
           dibatalkan.")
                 return slots,
    sudah_parkir, sudah_booking
                 else:
             print("Anda belum
        melakukan parkir.")
              return slots,
    sudah_parkir, sudah_booking
```

`confirmation = input(f"Apakah Anda ingin mengeluarkan mobil dari slot \033[31m{inp_slot}\033[0m? (Y/N): ")`

Meminta konfirmasi dari pengguna apakah mereka ingin mengeluarkan mobil dari slot yang sedang diparkir.

`if confirmation.upper() =='Y':`

Jika pengguna mengkonfirmasi (memilih 'Y'), maka status slot yang sedang diparkir diubah menjadi 'kosong', pesan sukses dicetak, dan history ditambahkan.

`slots[slot-1] = 'kosong'`

Mengubah status slot yang sedang diparkir menjadi 'kosong'.

`waktu_sekarang=datetime.now()`

Mendapatkan waktu saat ini.

`history.append(f"Anda keluar dari parkiran \033[31m{inp_slot}\033[0m pada {waktu_sekarang.strftime("%Y-%m-%d %H:%M")}")`

Menambahkan informasi ke dalam history bahwa mobil keluar dari parkiran pada waktu tertentu.

`sudah_booking = False`

Mengubah status sudah_booking menjadi False, karena mobil keluar dari parkiran.

`sudah_parkir = False`

Mengubah status sudah_parkir menjadi False, karena mobil keluar dari parkiran.

		`return slots, sudah_parkir, sudah_booking`
		Mengembalikan nilai slots, sudah_parkir, dan sudah_booking setelah melakukan operasi di atas.
		`elif confirmation.upper() == 'N':`
		Jika pengguna memilih untuk tidak mengeluarkan mobil, mencetak pesan bahwa pengeluaran mobil dibatalkan.
		`return slots, sudah_parkir, sudah_booking`
		Mengembalikan nilai slots, sudah_parkir, dan sudah_booking (tanpa perubahan) karena pengeluaran mobil dibatalkan.
		`else: print("Masukan tidak valid. Pengeluaran mobil dibatalkan.")`
		Jika input pengguna tidak valid (bukan 'Y' atau 'N'), mencetak pesan bahwa masukan tidak valid dan pengeluaran mobil dibatalkan.
		`else: print("Anda belum melakukan parkir.")`
		Jika mobil belum parkir, mencetak pesan bahwa pengguna belum melakukan parkir.
		`return slots, sudah_parkir, sudah_booking`
		Mengembalikan nilai slots, sudah_parkir, dan sudah_booking (tanpa perubahan) karena tidak ada operasi yang dilakukan jika mobil belum parkir.

`def timer_booking_timeout(slot,`	Waktu habis	Pertama-tama dilakukan deklarasi global variabel yaitu `global sudah_booking`

slot_parkir): global sudah_booking if slot_parkir[slot - 1] == 'booking' and sudah_booking: slot_parkir[slot - 1] = 'kosong' sudah_booking = False print(f"\nWaktu Booking untuk Slot \033[34m{slot}\033[0m Telah Habis.") thread = threading.Timer(30 ,timer_booking_timeout, args=(slot, slot_parkir)) thread.start()		yang memiliki fungsi menandai apakah proses booking sudah dilakukan atau belum. Ini digunakan untuk memastikan bahwa hanya satu slot yang dapat diubah statusnya ketika waktu habis. `if slot_parkir[slot - 1] == 'booking' and sudah_booking:` Merupakan pengecekan apakah slot parkir dengan nomor slot yang diberikan memiliki status `'booking'` dan variabel global `sudah_booking` bernilai `True`. Jika kondisi ini terpenuhi, maka berarti waktu booking telah habis. `slot_parkir[slot - 1] = 'kosong':` digunakan untuk mengubah status slot parkir menjadi `'kosong'`, menandakan bahwa waktu booking telah habis. Variabel global `sudah_booking` diubah menjadi `False`, menandakan bahwa waktu booking telah selesai. Setelah itu `print(f"\nWaktu Booking untuk Slot \033[34m{slot}\033[0m Telah Habis.")` digunakan untuk menampilkan pesan bahwa waktu booking untuk slot yang telah dibooking sebelumnya telah habis dengan menambahkan `\033[34m` untuk membuat kode slot parkir menjadi warna biru. `threading.Timer` disini berfungsi untuk melakukan hitung mundur di latar belakang, jadi untuk `thread = threading.Timer(30 ,timer_booking_timeout, args=(slot, slot_parkir)) thread.start()` berarti variabel thread akan melakukan hitung mundur selama `30` detik sebelum function `timer_booking_timeout` dijalankan. `args=(slot, slot_parkir)` memiliki fungsi yang sama untuk menggunakan variabel dari luar function
`history = []`	History	Pertama dilakukan deklarasi array kosong degnan nama `history`.

```
#saat parkir/masuk parkiran
waktu_sekarang=datetime.now()
history.append(f"Parkiran
\033[34m{inp_slot}\033[0m diisi
pada
{waktu_sekarang.strftime("%Y-%m-%d
%H:%M")}")

#mulai booking
waktu_sekarang = datetime.now()
waktu_selesai = datetime.now() +
timedelta(minutes=30)
history.append(f"Parkiran
\033[34m{inp_slot}\033[0m
dibooking pada
{waktu_sekarang.strftime("%Y-%m-%d
%H:%M")} dan akan berakhir pada
{waktu_selesai.strftime("%Y-%m-%d
%H:%M")}")

#batal booking
waktu_sekarang=datetime.now()
history.append(f"Booking slot
\033[31m{inp_slot}\033[0m
dibatalkan pada
{waktu_sekarang.strftime("%Y-%m-%d
%H:%M")}")

#keluar parkir
waktu_sekarang=datetime.now()
history.append(f"Anda keluar dari
parkiran \033[31m{inp_slot}\033[0m
pada
{waktu_sekarang.strftime("%Y-%m-%d
%H:%M")}")

#waktu booking habis
```

Untuk history dipisahkan menjadi beberapa jenis, yaitu:

1. **History ketika parkir**

Dilakukan pengambilan tanggal dan waktu saat dilakukan parkir dengan menggunakan `datetime.now()` dan disimpan ke dalam variabel `waktu_sekarang`. Untuk menampilkan waktu terkini, setiap jenis booking melakukan pengambilan waktu ulang.

Setelah itu dilakukan history.append untuk memasukkan string `f"Parkiran \033[34m{inp_slot}\033[0m diisi pada {waktu_sekarang.strftime("%Y-%m-%d %H:%M")}"`, yaitu akan menuliskan Parkiran {parkiran yang dipilih berwarna biru} diisi pada {waktu sekarang dengan format tahun-bulan-hari jam:menit)

`\033[34m` digunakan untuk mengubah teks setelah tanda tersebut menjadi biru, lalu `\033[0m` untuk mengembalikan teks berikutnya ke warna default.

`{waktu_sekarang.strftime("%Y-%m-%d %H:%M")}` digunakan untuk memanggil variabel `waktu_sekarang` lalu ditambahkan `.strftime` untuk mengatur format penulisan `waktu_sekarang`, yang dimana program kami menggunakan format `%Y`(untuk tahun)`-%m`(untuk bulan)`-%d`(untuk hari) `%H`(untuk jam)`:%M`(untuk menit).

2. **History ketika mulai booking**

Variabel `waktu_selesai` digunakan untuk menyimpan waktu booking habis, yaitu 30 menit setelah waktu mulai booking. Jadi digunakan `datetime.now() + timedelta(minutes=30)`.

Setelah itu sama seperti history sebelumnya digunakan `f"Parkiran \033[34m{inp_slot}\033[0m`

```
history.append(f"Waktu Booking slot \033[31m{inp_slot}\033[0m Telah Habis")

#show history
if history:
for i in range(0,len(history)):
print(f"{i+1}. {history[i]}")
os.system('cls')
else:
print("History kosong.")
```

dibooking pada {waktu_sekarang.strftime("%Y-%m-%d %H:%M")} dan akan berakhir pada {waktu_selesai.strftime("%Y-%m-%d %H:%M")}" untuk diappend ke dalam array history, yang membedakan adalah penambahan waktu berakhirnya booking pada bagian akhir history tetapi dengan cara dan format yang sama.

3. **History ketika batal booking**

Untuk batal booking digunakan cara yang sama yaitu `history.append(f"Booking slot \033[31m{inp_slot}\033[0m dibatalkan pada {waktu_sekarang.strftime("%Y-%m-%d %H:%M")}")`. Tetapi yang membedakan disini adalah warna kode slot parkirnya yaitu menggunakan `\033[31m` untuk mewarnai warna merah pada teks setelahnya lalu dikembalikan ke warna default dengan menggunakan `\033[0m`.

4. **History ketika keluar parkir**

Untuk keluar parkir hampir sama dengan append history sebelumnya, yang membedakan hanyalah kalimat yang dihasilkan. `history.append(f"Anda keluar dari parkiran \033[31m{inp_slot}\033[0m pada {waktu_sekarang.strftime("%Y-%m-%d %H:%M")}"`.menuliskan lebih ke keluar parkir tetapi sama dengan history ketika membatalkan booking, slot parkirnya menggunakan warna merah `\033[31m`.

5. **History ketika waktu booking habis**

Ketika waktu booking habis, program kami tidak menunjukkan tepat waktu habisnya karena sudah ditunjukkan pada history mulai booking.

`history.append(f"Waktu Booking slot \033[31m{inp_slot}\033[0m Telah Habis")`. Jadi yang ditunjukkan hanyalah informasi bahwa waktu booking slot yang dipilih sudah habis.

		6. Menampilkan list history Untuk menampilkan history ada dua kondisi, yaitu apakah ada history atau tidak ada history. Statement conditional yang digunakan adalah `if history:`, jika ada elemen di dalam history maka nilainya akan bersifat True dan akan melanjutkan perintah di dalamnya, yaitu `for i in range(0,len(history)): print(f"{i+1}. {history[i]}")`, yang dimana perintah ini berfungsi untuk menampilkan isi elemen history secara berurut. `for i in range(0,len(history)):` memiliki fungsi untuk melakukan perulangan agar semua history dapat ditampilkan. Syarat pada perulangan itu adalah ketika sudah mencapai jumlah history, maka iterasi akan berhenti, dan setiap kali perintah di dalamnya selesai dijalankan nilai `i` akan bertambah 1 sampai pada jumlah history yang ada. `print(f"{i+1}. {history[i]}")` ini yang digunakan untuk menampilkan list history. Oleh karena variabel i dimulai dari angka 0, maka untuk penomoran dilakukan `i+1`. Lalu history pada index `[i]` ditunjukkan yang dimana akan ditunjukkan satu per satu perulangan berhenti.

`available = [i + 1 for i, status in enumerate(slot_parkir) if status == 'kosong']`
 	
`print('='*29 + f"\n\x1b[37;1mSlot yang tersedia:\x1b[0m \033[34m{len(available)}/100\033[0m" + '\n' + '='*29)`
	Berapa Available	Pertama-tama untuk mencari jumlah parkiran yang kosong dilakukan pengecekan status pada array `slot_parkir`. Setelah itu diprint dengan menggunakan `f"\x1b[37;1mSlot yang tersedia:\x1b[0m \033[34m{len(available)}/100\033[0m"`, yang dimana untuk jumlah availablenya diberi warna abu-abu (`\x1b[37;1m`). Lalu untuk jumlah yang available digunakan perintah `len(available)`.
		
		Untuk `'='*29` hanya kami gunakan sebagai UI pada terminal agar rapi.

`#notifikasi booking`
`def notibooking():`
`popup = tk.Tk()`
	Notifikasi	**1. Notifikasi Booking**
		`popup = tk.Tk()` membuat window tkinter baru yang bernama `popup`. function

popup.wm_title("Parkirki' Notification")
label = tk.Label(popup, text="Anda telah sukses booking.")
label.pack(side="top", fill="x", pady=50)
 	
#notifikasi reminder
def notireminder():
global sudah_booking
if sudah_booking:
popup = tk.Tk()
popup.wm_title("Parkirki' Notification")
label = tk.Label(popup, text="Waktu booking Anda tersisa 15 detik.")
label.pack(side="top", fill="x", pady=50)
popup.mainloop()
 	
#waktu notifikasi reminder
threadreminder = threading.Timer(15 , notireminder)
threadreminder.start()
 	
#notifikasi parkir
def notiparked():
popup = tk.Tk()
popup.wm_title("Parkirki' Notification")
label = tk.Label(popup, text=f"Anda telah sukses parkir.")
label.pack(side="top", fill="x", pady=50)
		Tk digunakan untuk membuat window utama.
		
		`popup.wm_title("Parkirki' Notification")` mengubah popup window yang digunakan sebagai notifikasi menjadi "Parkirki' Notification".
		
		`label = tk.Label(popup, text="Anda telah sukses booking.")` digunakan untuk menambahkan label kepada popup yang bernama `label` dan mengatur teksnya menjadi "Anda telah sukses booking."
		
		`label.pack(side="top", fill="x", pady=50)` fungsi pack digunakan untuk mengatur posisi dan besar widget dalam window tkinter. paddy=50 mengatur agar label di letakkan di bagian atas window dengan padding 50px.
		
		2a. Notifikasi Reminder
		`popup = tk.Tk()` membuat window tkinter baru yang bernama `popup`. function Tk digunakan untuk membuat window utama.
		
		`popup.wm_title("Parkirki' Notification")` mengubah popup window yang digunakan sebagai notifikasi menjadi "Parkirki' Notification".
		
		`label = tk.Label(popup, text="Waktu Anda tersisa 15 detik.")` digunakan untuk menambahkan label kepada popup yang bernama `label` dan mengatur teksnya menjadi "Waktu booking Anda tersisa 15 detik."
		
		`label.pack(side="top", fill="x", pady=50)` fungsi pack digunakan untuk mengatur posisi dan besar widget dalam window tkinter. paddy=50 mengatur agar label di letakkan di bagian atas window dengan padding 50px.
		
		`popup.mainloop()` digunakan untuk menjalankan eventloop tkinter agar user bisa berinteraksi dengan window popup.

		2b. Waktu Notifikasi Reminder `threading.Timer(15 , notireminder)` membuat timer baru yang akan memanggil `notireminder` setelah 15 detik (setengah dari timeout booking). `threading.Timer` mengambil dua argumen: argumen pertama adalah berapa detik yang akan di tunggu sebelum memanggil function, dan argumen kedua adalah function yang akan di panggil. `threadreminder.start()` ini akan memulai timernya. **3. Notifikasi Parkir** `popup = tk.Tk()` membuat window tkinter baru yang bernama `popup`. function Tk digunakan untuk membuat window utama. `popup.wm_title("Parkirki' Notification")` mengubah popup window yang digunakan sebagai notifikasi menjadi "Parkirki' Notification". `label = tk.Label(popup, text="Anda telah sukses parkir.")` digunakan untuk menambahkan label kepada popup yang bernama `label` dan mengatur teksnya menjadi "Anda telah sukses parkir" `label.pack(side="top", fill="x", pady=50)` fungsi pack digunakan untuk mengatur posisi dan besar widget dalam window tkinter. paddy=50 mengatur agar label di letakkan di bagian atas window dengan padding 50px.

`def konversi_input_slot(input_slot):`
`  if len(input_slot) < 2 or not input_slot[0].isalpha() or not input_slot[1:].isdigit():`
`  #Memeriksa apakah panjang`
	Convert huruf inputan	`def konversi_input_slot`
		Membuat fungsi konversi_input_slot dengan satu parameter yaitu input_slot.

```
        input kurang dari 2 karakter,
      huruf pertama bukan huruf, atau
            sisanya bukan digit
                return -1 #return -1
    supaya ini jadi tidak valid karena
            tidak ada di slot

            row = ord(input_slot[0]) -
                    ord('A')
            col = int(input_slot[1:])

        if row < 0 or row >= 10 or col
                < 1 or col > 10:
            #Memeriksa apakah baris di
    luar rentang A-J dan kolom di luar
                rentang 1-10
                return -1  #return -1
    supaya ini jadi tidak valid karena
            tidak ada di slot
```

```
if len(input_slot) < 2 or not
input_slot[0].isalpha() or not
input_slot[1:].isdigit():
```

Memeriksa kondisi apakah panjang input kurang dari 2 karakter, huruf pertama bukan huruf, atau sisanya bukan digit.

```
len(input_slot) < 2
```

Panjang input kurang dari 2 karakter.

```
not input_slot[0].isalpha()
```

Huruf pertama bukan huruf.

```
not input_slot[1:].isdigit():
```

Sisanya bukan digit (dari karakter kedua hingga akhir).

```
return -1
```

Mengembalikan nilai -1 sebagai indikasi bahwa input tidak valid karena tidak memenuhi kondisi di atas.

```
row = ord(input_slot[0]) -
ord('A')
```

Mengonversi huruf pertama dari input menjadi nilai ASCII dan mengurangkan dengan nilai ASCII huruf 'A'. Hal ini dilakukan untuk mendapatkan nilai baris dalam sistem koordinat.

```
col = int(input_slot[1:])
```

Mengonversi sisanya setelah huruf pertama menjadi integer. Hal ini dilakukan untuk mendapatkan nilai kolom dalam sistem koordinat.

```
if row < 0 or row >= 10 or col
< 1 or col > 10:
```

Memeriksa apakah nilai baris berada di luar rentang A-J (0-9) atau nilai kolom di luar rentang 1-10.

```
row < 0
```

Nilai baris kurang dari 0.

		`row >= 10` Nilai baris lebih besar atau sama dengan 10. Nilai kolom kurang dari 1. **`col < 1`** Nilai kolom kurang dari 1. **`col > 10`** Nilai kolom lebih besar dari 10. **`return -1`** Mengembalikan nilai -1 sebagai indikasi bahwa input tidak valid karena nilai baris atau kolom berada di luar rentang yang diperbolehkan.