MODUL 2

Closures dan Scope Variabel

Praktikum Pemrograman Berbasis Fungsi

SAINS DATA
INSTITUT TEKNOLOGI SUMATERA

2.1. Tujuan

1. Mahasiswa memahami perbedaan scope variable global dan lokal.
2. Mahasiswa mampu menerapkan closure dalam program sederhana

2.2. Konsep Dasar

2.2.1. Variabel

Variabel adalah tempat penyimpanan data yang digunakan untuk menyimpan nilai yang dapat berubah selama program berjalan. Namun, aksesibilitas variabel dalam suatu program bergantung pada ruang lingkup (scope) variabel tersebut. Variabel digunakan untuk merujuk ke suatu nilai dalam memori komputer. Dalam Python tidak perlu mendeklarasikan tipe variabel karena bersifat dynamically typed (tipe data ditentukan secara otomatis).

```
1    x = 10
2    nama = "Budi"
```

Gambar 2.1. contoh variable

Pada gambar 2.1 dapat dilihat bahwa x menyimpan nilai 10 dan variable nama menyimpan string “Budi”. Setiap variabel memiliki ruang lingkup (scope) yang menentukan di mana variabel tersebut dapat diakses atau diubah dalam code. Ruang lingkup variabel menentukan cakupan di mana variabel tersebut dapat digunakan dalam kode program. Ada tiga jenis ruang lingkup variabel:

1. Variabel Lokal (Local Variable)
2. Variabel Global (Global Variable)
3. Variabel Nonlocal (Nonlocal Variable

Pengelolaan variabel dan ruang lingkup dalam pemrograman berbasis fungsi dimana mempengaruhi bagaimana data disimpan dan digunakan di dalam fungsi. Dalam pemrograman berbasis fungsi, variabel dikelola dengan menghindari perubahan state global dan lebih banyak menggunakan variabel lokal serta konsep closure untuk mempertahankan state dalam fungsi yang bersifat immutabel.

2.2.2. Local Variabel

Variabel yang dideklarasikan di dalam suatu fungsi dan hanya bisa digunakan di dalam fungsi tersebut disebut local variabel. Jika mencoba mengakses variabel lokal dari luar fungsi, maka akan terjadi error. Pada gambar 2.2. dapat dilihat bahwa variabel local yaitu x hanya bisa diakses didalam fungsi contoh_local. Dan akan menghasilkan error saat variabel x diakses di luar fungsi pada baris ke 5-6 seperti pada gambar.

```
1  def contoh_local():
2      x = 5
3      print("Nilai x di dalam fungsi:", x)
4
5  contoh_local()
6  print(x)
7
```

Gambar 2.2. contoh variabel local

Dalam pemrograman berbasis fungsi, variabel lokal sering digunakan untuk menghindari efek dengan variabel di luar fungsi, Membantu menciptakan fungsi yang pure (tidak bergantung pada state luar).

2.2.3. Global Variabel

Variabel yang dideklarasikan di **luar fungsi** dan dapat digunakan di seluruh program, termasuk di dalam fungsi. Namun, dalam pemrograman berbasis fungsi, penggunaan variabel global sebaiknya dihindari karena menyebabkan efek samping dan membuat fungsi tidak murni (impure function). Keuntungan dari variable global adalah bisa digunakan di dalam dan luar fungsi. Namun kekurangannya yaitu mengurangi modularitas dan bisa menyebabkan bug karena perubahan state global.

```
9   y = 10
10
11  def contoh_global():
12      print("Nilai y di dalam fungsi:", y)
13
14  contoh_global()
15  print("Nilai y di luar fungsi:", y)
```

Gambar 2.3. contoh global variable

Pada gambar 2.3 dapat dilihat contoh dari global variable. Pada baris ke 9, y merupakan variable global dimana nilai y dapat diakses didalam fungsi contoh_global dan dapat diakses di fungsi lain pada bari ke 14 dan 15. Jika nilai variable ingin dirubah dari fungsi lain maka gunakan keyword global seperti pada gambar 2.4.

```
17  z = 10
18
19  def ubah_global():
20      global z
21      z = 200
22      print("Nilai z di dalam fungsi:", z)
23
24  ubah_global()
25  print("Nilai z di luar fungsi:", z)
```

Gambar 2.4 mengubah variable global

2.2.4. Non Local Variabel

Variabel nonlocal digunakan dalam nested function (fungsi dalam fungsi) untuk merujuk ke variabel dari fungsi luar tanpa mengubah variabel global. Dalam

pemrograman berbasis fungsi, nonlocal digunakan ketika ingin mempertahankan state lokal dalam closure tanpa menggunakan variabel global. Keuntungan dari variable non local adalah state variable tetap tersimpan tanpa perlu variabel global namun variable non local hanya bisa diakses melalui fungsi dalam.

```
31  def pembuat_counter():
32      hitungan = 0
33
34      def tambah():
35          nonlocal hitungan
36          hitungan += 1
37          return hitungan
38
39      return tambah
40
41  counter1 = pembuat_counter()
42  print(counter1())
43  print(counter1())
44  print(counter1())
```

Gambar 2.5. contoh variable local.

2.2.5. Closure

Closure adalah konsep dalam pemrograman berbasis fungsi yang memungkinkan sebuah fungsi mengakses variabel dari lingkup luar meskipun fungsi tersebut dieksekusi di luar lingkup aslinya. Closure sering digunakan untuk menyimpan state tanpa menggunakan variabel global. Closure terjadi ketika sebuah inner function (fungsi dalam fungsi) masih dapat mengakses variabel dari outer function (fungsi luar), meskipun outer function sudah selesai dieksekusi.

Karakteristik pemrograman berbasis fungsi adalah pure function yaitu fungsi yang tidak mengubah state luar dan selalu menghasilkan output yang sama untuk input yang sama sehingga fungsi yang menggunakan atau mengubah variabel di luar fungsi sehingga outputnya bisa berubah-ubah tergantung state luar.

```
29  # Contoh Pure Function
30  def kali_dua(x):
31      return x * 2
32
33  print(kali_dua(5))  # Output: 10
34
35  # Contoh Impure Function
36  nilai_awal = 3
37
38  def tambah_nilai(x):
39      return x + nilai_awal
40
41  print(tambah_nilai(5))
```

Gambar 2.5. perbandingan pure fungsi dan impure fungsi.

Closure bekerja dengan cara menyimpan referensi ke variabel dari fungsi luar yang telah selesai dieksekusi. Variabel ini tetap ada dalam memori dan bisa diakses oleh fungsi dalam yang dikembalikan.

```
45  def pembuat_pangkat(n):  # Fungsi luar
46      def pangkat(x):  # Fungsi dalam (closure)
47          return x ** n  # Menggunakan variabel dari fungsi luar
48      return pangkat  # Mengembalikan fungsi dalam sebagai objek
49
50  kuadrat = pembuat_pangkat(2)
51  kubik = pembuat_pangkat(3)
52
53  print(kuadrat(4))
54  print(kubik(4))
```

Gambar 2.6. contoh closure

Pada gambar dapat dilihat fungsi pembuat_pangkat(2) mengembalikan fungsi pangkat(x) yang menyimpan n = 2, kemudian fungsi kuadrat(4) memanggil fungsi pangkat(x) dengan x = 4, hasilnya 16 dan begitupun fungsi kubik. Closure memungkinkan kita membuat fungsi yang menyimpan konfigurasi awal tanpa perlu variabel global.

Closure digunakan saat:

a. Menghindari penggunaan variabel global.
b. Membuat fungsi yang dapat menyimpan state.
c. Menggunakan konfigurasi awal tanpa harus terus menerus meneruskan parameter.
d. Membantu dalam teknik functional programming seperti currying dan decorators.

Kelebihan	**Kekurangan**
Mengurangi penggunaan variabel global	Bisa menyebabkan konsumsi memori lebih besar
Memudahkan modularisasi kode	Sulit dipahami oleh pemula
Memungkinkan penyimpanan state	Bisa menyulitkan debugging jika tidak digunakan dengan baik
Berguna dalam functional programming	Bisa menyebabkan memory leak jika tidak dikelola dengan baik

1. Prosedur Percobaan

Kasus :

Seorang developer game ingin membuat program dengan fitur penghitung langkah sederhana yang bisa digunakan untuk melacak jumlah langkah karakter dalam game. Fitur yang harus ada dalam program adalah menambah jumlah langkah yang telah ditempuh, melihat total langkah yang sudah dikumpulkan, mereset jumlah langkah ke nol jika pengguna ingin memulai dari awal.

Solusi :

```python
def buat_penghitung_langkah():
    tambah_langkah = lambda langkah_sekarang, jumlah: langkah_sekarang + jumlah
    lihat_total = lambda langkah_sekarang: "Total langkah saat ini: " + str(langkah_sekarang)
    reset = lambda: 0  # Mengembalikan nilai awal tanpa mengubah state luar

    return tambah_langkah, lihat_total, reset


tambah, lihat, reset = buat_penghitung_langkah()

langkah = 0

langkah = tambah(langkah, 100)
print(lihat(langkah))

langkah = tambah(langkah, 50)
print(lihat(langkah))

langkah = reset()
print(lihat(langkah))
```

2. Tugas Individu