λ MODUL

PRAKTIKUM PEMROGRAMAN BERBASIS FUNGSI

Ardika Satria, M.Si
Ahmad Luky R. M.Kom
Yuliana, M.Cs
Ade Lailani, M.Si

SAINS DATA
INSTITUT TEKNOLOGI SUMATERA

MODUL 1

Pengantar Pemrograman Berbasis Fungsi

Praktikum Pemrograman Berbasis Fungsi

SAINS DATA
INSTITUT TEKNOLOGI SUMATERA

1.1 Tujuan

1. Memahami pengenalan bahasa pemrograman python berbasis fungsi
2. Memahami konsep dasar paradigma pemrograman berbasis fungsi

1.2 Konsep Dasar

1.2.1. **Fungsi**

Fungsi adalah blok kode yang hanya berjalan ketika dipanggil. Fungsi dapat meneruskan data, yang dikenal sebagai parameter, ke dalam suatu fungsi. Sebuah fungsi dapat mengembalikan data sebagai hasilnya. Fungsi terdiri dari input , process dan output. Input dari sebuah fungsi dalam python disebut argumen/parameter, sedangkan output disebut dengan return value. Fungsi didefinisikan dengan keyword def, dan dapat dipanggil dalam program utama untuk mengeksekusi blok kode yang telah didefinisikan tersebut. Blok kode tersebut adalah function body nya. Function body adalah process yang dilakukan oleh fungsi saat fungsi itu dipanggil. Perhatikan gambar 1.1.

```
def square(x):
  y = x*x
  return y
```

Gambar 1.1. contoh fungsi

Pada gambar dapat dilihat sebuah fungsi dengan nama square. Fungsi square memiliki input parameter/argumen yaitu variabel x. Proses Assignment y=x*x adalah function body dari square. Output dari fungsi square adalah y, sebagai hasil kuadrat dari input x. Keyword return menandakan ekspresi yang dikeluarkan oleh fungsi tersebut sebagai output.

1.2.2. Pemrograman Berbasis Fungsi

Pemrograman fungsional adalah paradigma yang memandang komputasi sebagai evaluasi fungsi matematika dan menghindari perubahan keadaan (state) serta data yang dapat dimodifikasi. Karakteristik utama pemrograman fungsional adalah:

1. Immutability
   Immutablity adalah data yang tidak bisa diubah setelah dibuat. Jika ingin memodifikasi data harus membuat salinan baru daripada mengubah data yang sudah ada. Imutability dapat digunakan untuk menghindari perubahan tak terduga dalam program, karena data tidak bisa berubah, sehingga aman digunakan dalam program multi-thread selain itu juga lebih mudah untuk debugging tidak perlu mengetahui siapa yang mengubah nilai. Gambar 1.2. merupakan contoh imutabile data yauti string dan python.

```
s = "Halo"
s = s + " Dunia"
print(s)


t = (1, 2, 3)
t_baru = t + (4, 5)
print(t_baru)
```

Gambar 1.2

2. First-class functions

   Sebuah first-class object dalam bahasa pemrograman berarti objek yang dapat diperlakukan seperti nilai lain (misalnya angka atau string). Jika fungsi dianggap sebagai first-class object, itu berarti fungsi bisa:

   1. Disimpan dalam variable
   2. Dikirim sebagai argumen ke fungsi lain
   3. Dikembalikan sebagai nilai dari fungsi lain
   4. Disimpan dalam struktur data (list, dictionary, dsb.)

   Contoh fungsi fungsi sebagai first class object dapat dilihat pada gambar 1.3.

```
def salam():
    return "Halo, dunia!"

ucapan = salam
print(ucapan())


def panggil_fungsi(fungsi):
    return fungsi()

print(panggil_fungsi(salam))

def pilih_operasi(operator):
    if operator == "tambah":
        return lambda x, y: x + y
    elif operator == "kali":
        return lambda x, y: x * y

operasi = pilih_operasi("tambah")
print(operasi(4, 5))

fungsi_dict = {
    "halo": salam,
    "tambah": lambda x, y: x + y
}

print(fungsi_dict["halo"]())
print(fungsi_dict["tambah"](3, 4))
```

3. Rekursi
   Rekursi adalah cara umum dalam pemrograman fungsional untuk menggantikan loop.

```python
def faktorial(n):
    if n == 0:
        return 1
    return n * faktorial(n - 1)

print(faktorial(5))


def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n - 1) + fibonacci(n - 2)

print(fibonacci(6))
```

4. Pure functions
   Pure functions adalah fungsi yang selalu menghasilkan output yang sama untuk input yang sama dan tidak memiliki efek samping (side effects). Pure function hanya bergantung pada inputnya sendiri dan tidak mengubah variabel di luar fungsi.

pertemuan2b.py > ...

```python
def tambah(a, b):
    return a + b

print(tambah(3, 5))
print(tambah(3, 5))


total = 0

def tambah_ke_total(a):
    global total
    total += a
    return total

print(tambah_ke_total(3))
print(tambah_ke_total(3))
```

1.2.3. Evaluasi Ekspresi

Evaluasi ekspresi adalah proses menghitung nilai dari suatu ekspresi berdasarkan aturan bahasa pemrograman atau sistem matematika. Evaluasi ini terjadi ketika kita menjalankan suatu ekspresi dalam program atau dalam sistem formal seperti Lambda Calculus.

```
71  x = 3 + 4 * 2
72  print(x)
73
```

Evaluasi yang terjadi:
4 * 2 → 8
3 + 8 → 11
Komputer akan menghitung hasilnya secara bertahap sesuai dengan aturan operator.

1.2.4. Anonymous Fungsi

Anonymous function adalah fungsi tanpa nama yang dideklarasikan secara langsung tanpa menggunakan kata kunci `def` (dalam Python). Di Python, anonymous function disebut sebagai lambda function. Lambda function sering digunakan untuk operasi sederhana dan langsung diterapkan pada data. Lambda fungsi digunakan jika hanya memiliki satu ekspresi sederhana. Berikut adalah contoh dari lambda function:

```
80  perkalian = lambda x, y: x * y
81  print(perkalian(5, 4))
```

```
def perkalian(x, y):
    return x * y
```

```
lambda arguments: expression
```

Keterangan :

1. lambda → Kata kunci untuk mendefinisikan fungsi lambda.
2. arguments → Parameter yang akan diterima fungsi.
3. expression → Operasi yang akan dilakukan dan hasilnya dikembalikan secara otomatis.
4. Lambda hanya bisa berisi satu ekspresi sehingga tidak bisa memiliki beberapa pernyataan atau loop
5. Lambda mengembalikan hasil ekspresi tanpa menggunakan return

1.3 Prosedur Percobaan

Kasus :Pembuatan Sistem Diskon di E-Commerce

Sebuah e-commerce ingin menghitung harga akhir produk setelah diberikan diskon berdasarkan tipe pelanggan:

1. Pelanggan Reguler → Diskon 5%
2. Pelanggan VIP → Diskon 10%
3. Pelanggan Premium → Diskon 20%

Solusi :

Berdasarkan kasus yang diberikan gunakan fungsi sebagai parameter untuk menghitung diskon. Pada gambar digunakan fungsi hitung_diskon(), fungsi hitung_diskon() merupakan

implementasi dari First-Class Function yaitu mengembalikan fungsi lambda. Selanjutnya menggunakan dictionary, dimana dictionary DISKON tidak berubah setelah dibuat yang merupakan implemntasi dari Immutability. Selanjutnya mengimplementasikan Evaluasi Ekspresi yaitu ungsi hanya dipanggil saat diperlukan.

```
1   def hitung_diskon(pelanggan):
2       return lambda harga: harga * (1 - pelanggan)
3
4
5   DISKON = {
6       "reguler": hitung_diskon(0.05),
7       "vip": hitung_diskon(0.10),
8       "premium": hitung_diskon(0.20)
9   }
10
11
12  def harga_akhir(harga, tipe_pelanggan):
13      return DISKON.get(tipe_pelanggan, lambda x: x)(harga)
14
15
16  print(harga_akhir(100000, "reguler"))
17  print(harga_akhir(200000, "vip"))
18  print(harga_akhir(500000, "premium"))
19  print(harga_akhir(150000, "baru"))
```

Pada gambar juga dapat dilihat bahwa implementasi pure function sudah digunakan.

```
1   def hitung_diskon(pelanggan):
2       return lambda harga: harga * (1 - pelanggan)
3
```