Modul Pemrograman Dasar

Sevi Nurafni

Catatan

1. Modul ini dirancang untuk dapat menjadi pegangan mata kuliah Pemrograman Dasar
2. Anda dapat membuka modul ini saat latihan praktikum.
3. Anda sangat disarankan untuk mencoba menjalankan semua program modul ini di komputer Anda, supaya Anda dapat mengetahui keluaran dari program yang ada.
4. Anda sangat disarankan untuk bereksperimen dari program-program yang ada di modul ini supaya Anda mendapat gambaran lebih jelas mengenai apa yang program Anda lakukan.
5. Anda sangat disarankan membaca tutorial dari tempat lain dan mengeksplor sendiri bahasa yang Anda gunakan.

1 Modul 1

1.1 Pendahuluan

Pada modul ini, kita menggunakan Python versi 3.9. Anda dapat mendownload Python di https: //www.python.org/downloads/release/python-390/, pilihan ”Windows x86-64 executable in- staller”.

Beberapa karakteristik dari bahasa ini:

- Python bersifat *case-sensitive*, artinya perbedaan huruf besar dan huruf kecil menyebabkan perbedaan makna.
- Python sangat memperhatikan indentasi dan pergantian baris. Kesalahan indentasi dapat menyebabkan gagal *compile* hingga kesalahan program.
- Variabel di python bersifat implisit dan dinamis. Artinya, sebuah variabel tidak perlu dideskripsikan tipe datanya. Namun di modul ini kita tetap mempelajari tipe data yang ada.

1.2 Input dan Output

Dalam python, program untuk menulis "Hello, World!" ke layar adalah seperti berikut:

```
print(" Hello , World!")                    #   (1)

print(" Ini program", end="")               #   (2)
print(" pertama saya")                      #   (3)

print(" Dengan"," Koma")
# Ini akan menghasilkan luaran " Dengan Koma"
print(" Dengan" + " Plus")
# Ini akan menghasilkan luaran " DenganPlus",
# Perhatikan '+' pada print melakukan konkatinasi pada string saja.

variabel = " sebuah nilai"
print(f" Dengan format { variabel}")
# Ini akan menghasilkan luaran " Dengan format sebuah nilai"
# { variabel} akan diubah dengan nilai dari variabel bersesuaian.

# Ini adalah sebuah komentar.

# Semua yg ada setelah tanda pagar (#) # akan
diabaikan oleh interpreter.

  """
  Selain itu , kita juga bisa membuat komentar multi baris dengan
tiga petik (""")
  """
```

Bagian yang ditandai nomor 1 bertugas menuliskan "Hello, World!" ke layar. Sintaks print akan otomatis memindahkan baris setelah selesai menulis ke layar, kecuali disertai parameter end="". Berarti, output program di atas adalah "Hello, World!" diikuti dengan "Ini program pertama saya " di baris selanjutnya.

Untuk melakukan input, kita membutuhkan penampung untuk menyimpan data yang diin- putkan. Sebagai contoh, di bawah ini adalah program yang menerima input dan menuliskan ulang yang dimasukkan.

```
S = input(" Masukkan kalimat: ")                                    #  1
print(" Anda memasukkan kalimat " + S)                              #  2

N = int( input(" Masukkan sebuah angka: "))                         #  3
print(" Jika angka Anda ditambah 5, hasilnya " + str(N + 5))        #  4
```

Pada bagian nomor (1) dan (3), program membaca input dari user dan dimasukkan ke variabel S dan N. Lalu, bila ingin program menerima sebuah bilangan (sehingga bisa dijumlah / dikali / lainnya), kita bungkus input() dengan int(). Selain int, string (kumpulan karakter), Python dapat menerima tipe data float (bilangan real), dan masih banyak lagi.

1.3 Tipe Data

Ada beberapa macam tipe data, namun dalam Pengenalan Komputasi kita hanya akan banyak menggunakan tipe data berikut:

bool	Boolean	True atau False
int	Bilangan bulat	seluruh bilangan bulat
float	Bilangan real	seluruh bilangan real
string	Teks	kumpulan karakter

Contoh penggunaan:

```
B = True                    # Boolean
I = 123456789012 # Bilangan bulat
R = 2.331973          # Bilangan real
S = " abc" # Teks
S = 'def'
```

Perhatkan pada variabel S, Anda bebas menggunakan petik satu (') atau petik dua (")

1.4 Operator

1.4.1 Operator Aritmatika

Operator	Deskripsi	Contoh
+	Penjumlahan	2 + 3 bernilai 5
-	Pengurangan	1 - 8 bernilai -7
*	Perkalian	5 * 6 bernilai 30
/	Pembagian	13 / 5 bernilai 2.6
//	Pembagian (dibulatkan ke bawah)	13 // 5 bernilai 2
%	Sisa Bagi / Modulo	13 % 5 bernilai 3

1.4.2 Operator Assignment

Operator	Deskripsi	Contoh
=	Assignment	N = 5
+=	Penjumlahan	N += 5, N akan ditambah 5.
-=	Pengurangan	N -= 5, N akan dikurang 5.
*=	Perkalian	N *= 5, N akan dikali 5.
/=	Pembagian	N /= 5, N akan dibagi 5.
//=	Pembagian (dibulatkan ke bawah)	N //= 5, N akan dibagi 5 (dibulatkan ke bawah).
%=	Sisa Bagi / Modulo	N %= 5, N akan dimodulo 5.

1.4.3 Operator Relasional

Operator	Deskripsi	Contoh True	Contoh False
==	Sama dengan	2 == 2	2 == 3
!=	Tidak sama dengan	3 != 2	3 != 3
<	Kurang dari	2 < 3	2 < 2
>	Lebih dari	3 > 2	2 > 3
<=	Kurang dari sama dengan	2 <= 2	3 <= 1
>=	Lebih dari sama dengan	6 >= 5	2 >= 4

1.4.4 Operator Logika

Operator	Deskripsi	Contoh True	Contoh False
and	Dan: True jika kedua operand True	(1 < 2)and (3 == 3)	(1 == 2)and (3 == 3)
or	Atau: True jika salah satu operand True	(1 < 2)or (4 == 3)	(3 < 2)or (2 == 3)
not	Negasi: True jika operand False	not (3 < 2)	not (1 < 2)

1.5 Percabangan

Dalam pemrograman, terdapat percabangan. Dengan demikian, program kita dapat berperilaku tergantung input user. Misal kita buat program yang memeriksa apakah sebuah bilangan positif:

```
print(" Masukkan nilai N: ", end="")
N = int( input())

if (N > 0):
    print( str(N) + " bilangan positif")

        # str(N) berfungsi mengubah N menjadi string agar dapat dikonkatenasi saat print
```

Lalu, jika kita ingin menuliskan kebalikannya:

```
  ...
if (N > 0):
   print( str(N) + " bilangan positif")
   else: # N <= 0
```

```
    print( str(N) + " bilangan bukan positif")
...
```

Namun, kita tahu kadang bilangan bisa nol atau negatif, jadi perlu kita tambahkan:

```
...
if (N > 0):
      print( str(N) + " bilangan positif")
elif (N < 0):
      print( str(N) + " bilangan negatif")
else: # N == 0
      print( str(N) + " bilangan nol")
...
```

Perhatikan juga kalau kita bisa membuat else ini berulang sampai yang kita mau. Selain itu, kita juga bisa meletakkan if di dalam if.

```
...
if (N >= 0):
          if (N > 0):
              print( str(N) + " bilangan positif")
           else: # N == 0
              print( str(N) + " bilangan nol")
else: # N < 0
          print( str(N) + " bilangan negatif")
...
```

1.6 Perulangan

Pada pemrograman, sering kali dibutuhkan pemrosesan berulang-ulang untuk mencapai suatu hasil tertentu. Apabila pengulangan ini dilakukan secara manual ukuran file program akan menjadi terlalu besar. Contoh sederhana adalah jika kita ingin menuliskan “Hello World” di layar sebanyak 1000 kali, maka akan dibutuhkan paling tidak 1000 baris perintah. Menggunakan sintaks pengulangan, persoalan tersebut dapat diselesaikan hanya menggunakan beberapa baris program.

```
    for i in range (1000):
        print(" Hello World")
```

1.1.1 While Loop

Salah satu sintaks yang looping / pengulangan yang sering digunakan adalah sintaks While-Do. Program akan mengecek sebuah kondisi yang diberikan terlebih dahulu sebelum menjalankan statement yang ada di dalamnya.

Berikut adalah program yang menerima *a* dan *b* dan menuliskan *a*, *a* + 1, *a* + 2, . . . , *b* − 1, *b*.

```
a = int( input())
b = int( input())
 i = a
while (i <= b):
      print(i)
      i += 1
```

Perhatikan bahwa:

1. Sintaks while loop adalah **while (kondisi):**

2. Perulangan body while loop akan terus dilakukan selama kondisi terpenuhi.
3. Body while adalah bagian kode yang indentasi ke dalam body while loop.

1.6.1 FOR LOOP

Bentuk looping yang kedua adalah bentuk For. Bentuk ini umumnya digunakan untuk pen- gulangan yang sudah diketahui jumlahnya. Namun, for loop juga dapat dibuat menggunakan while-do loop.

Berikut adalah program yang menerima *a* dan *b* dan menuliskan *a*, *a* + 1, *a* + 2, . . . , *b* − 1, *b*.

```
a = int( input())
b = int( input())
for i in range(a, b+1):
    print(i)
```

Perhatikan bahwa:

1. Sintaks for loop adalah **for variable in range(value):**
2. Perulangan body for loop akan dilakukan sebanyak jumlah range yang diberikan.
3. Body for loop adalah bagian kode yang indentasi ke dalam body for loop.
4. Nilai variable akan berubah sesuai dengan nilai selanjutnya dalam range setiap perulangan.

range dapat digunakan dengan satu, dua, atau tiga parameter. Untuk lebih jelasnya, perhatikan potongan kode berikut:

```
range(10)          # 0, 1 , 2, 3, 4, 5, 6, 7, 8, 9
range(2,5)         # 2, 3, 4
range(7, 2, -1)    # 7, 6, 5, 4, 3
range(2, 8, 2)     # 2, 4, 6
```

1.6.2 Perulangan Bersarang

Perhatikan pula, perulangan dapat dilakukan di dalam perulangan. Sebagai contoh, berikut adalah program untuk membuat pola persegi

```
n = int( input())
for i in range(n):
    for j in range(n):
        print("*", end="") # Dieksekusi sebanyak n * n kali
print() # Dieksekusi sebanyak n kali
# Perhatikan indentasai di dalam perulangan.
```

1.7 Latihan Modul 1

1.7.1 Buatlah sebuah program sapaan dalam notasi algoritma dengan spesifikasi sebagai berikut:

- menampilkan tulisan “Halo, siapa namamu?” di layar, lalu
- meminta pengguna memasukan namanya, lalu
- menampilkan tulisan “Di kota apa kamu sekarang?” di layar, lalu
- meminta pengguna memasukkan nama kotanya sekarang, dan akhirnya
- menuliskan “Senang berteman denganmu <nama>, di kota <kota>

1.7.2 Jumlah penduduk di suatu negara adalah jumlah kelahiran ditambah jumlah yang berimigrasi ke negara tersebut lalu dikurangi dengan jumlah kematian dan jumlah yang beremigrasi ke negara lain. buatkan program untuk menghitung jumlah penduduk di suatu negara pada tahun tertentu

1.7.3 Seorang kasir di sebuah toko ingin membuat program untuk menghitung total harga belanjaan pelanggan. Program harus meminta pengguna untuk memasukkan jumlah barang yang dibeli. Untuk setiap barang, pengguna harus memasukkan harga satuannya. Gunakan **perulangan** untuk meminta input harga setiap barang, lalu hitung total harga belanjaan. Jika total harga belanjaan lebih dari Rp100.000, berikan diskon 10%. Jika tidak, harga tetap. Terakhir, gunakan **pemilihan (if-else)** untuk menentukan apakah pelanggan mendapatkan diskon atau tidak, dan cetak total harga akhir setelah diskon (jika ada).