Modul Pemrograman Lanjut

Sevi Nurafni

Catatan

1. Modul ini dirancang untuk dapat menjadi pegangan mata kuliah Pemrograman Lanjut
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 pemrograman Python:

- Python bersifat *case-sensitive*, artinya perbedaan huruf besar dan huruf kecil menyebabkan perbedaan makna sintaks.
- Python sangat memerhatikan *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 konkatenasi 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 yang 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  # Bilangan Boolean
I = 123456789012  # Bilangan bulat
R = 2.331973  # Bilangan real
S = "abc"  # Teks
S = 'def'  # Mengganti nilai teks S menjadi 'def'
```

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

1.4 Operator

Operator dalam Python adalah simbol atau kata kunci yang digunakan untuk melakukan operasi pada nilai (operand). Operator berfungsi untuk melakukan perhitungan matematika, manipulasi data, atau pengambilan keputusan dalam program. Berikut beberapa jenis operator dalam Python:

1.4.1 Operator Aritmatika

Digunakan untuk operasi matematika.

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 Penugasan)

Mengatur atau memperbarui nilai variabel.

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 Perbandingan)

Membandingkan dua nilai.

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 bernilai TRUE.	(1 < 2) and (3 == 3)	(1 == 2) and (3 == 3)
or	Atau: TRUE, jika salah satu operand bernilai TRUE.	(1 < 2) or (4 == 3)	(3 < 2) or (2 == 3)
not	Negasi: TRUE, jika operand bernilai 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) + " adalah 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) + " adalah bilangan positif")
else:  # N <= 0
   print(str(N) + " adalah bilangan bukan positif"

...
```

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

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

1.6 Latihan Modul 1

1.6.1 Program Ganjil-Genap

Menggunakan contoh program bilangan positif, negatif, nol sebelumnya, kembangkanlah pro- gram tersebut sehingga ketika masukan bilangan positif program akan menentukan bilangan tersebut ganjil atau genap.

Contoh 1

```
Masukkan nilai N: 2
2 bilangan positif genap
```

1.6.2 Kalkulator Sederhana

Buatlah sebuah program kalkulator sederhana yang menerima 2 buah angka dan sebuah karakter operasi, dan menuliskan hasil perhitungannya. Operator yang diterima adalah + (tambah), - (kurang), * (kali), / (bagi, dibulatkan ke bawah), % (sisa bagi).

Contoh 2

```
Masukkan angka pertama: 2
Masukkan angka kedua: 6
Masukkan operator: +

Output:
2 + 6 = 8
```

Contoh 3

```
Masukkan angka pertama: 13
Masukkan angka kedua: 5 Masukkan
operator: %
  13 % 5 = 3
```