Memahami Algoritma dan Pemrograman

Sebelum menulis kode, penting untuk memahami hubungan antara algoritma dan pemrograman.

Algoritma: Ini adalah serangkaian langkah atau instruksi yang terstruktur untuk memecahkan sebuah masalah.

Pemrograman: Ini adalah proses mengubah algoritma menjadi bahasa yang bisa dipahami oleh komputer, yaitu kode program.

Jadi, langkahnya adalah: Masalah --> Algoritma --> Kode Program

Apa Itu Bahasa Pemrograman?

Bahasa pemrograman adalah serangkaian instruksi yang kita berikan kepada komputer untuk menjalankan suatu tugas. Anggap saja seperti bahasa yang digunakan untuk berbicara dengan komputer. Python adalah salah satu bahasa tersebut, yang dirancang agar mudah dipahami oleh manusia.

Bahasa pemrograman itu ada banyak sekali, dan setiap bahasa punya keunggulan serta tujuan penggunaannya masing-masing.

Berikut adalah beberapa contoh bahasa pemrograman yang paling umum dan sering dipakai:

- **Python**: Sangat populer karena mudah dipelajari. Cocok untuk membuat website, analisis data, dan kecerdasan buatan (AI).
- **JavaScript**: Bahasa utama untuk membuat halaman website menjadi interaktif, seperti membuat animasi atau memvalidasi formulir.
- **Java**: Kuat dan stabil, sering dipakai untuk membuat aplikasi Android dan sistem berskala besar di perusahaan.
- **C++**: Bahasa yang sangat cepat. Digunakan untuk membuat game, sistem operasi, dan aplikasi yang butuh performa tinggi.
- **C#** (C Sharp): Dikembangkan oleh Microsoft, ini adalah pilihan utama untuk membuat game menggunakan engine Unity dan aplikasi desktop Windows.
- **PHP**: Bahasa yang dirancang khusus untuk pengembangan web di sisi server. Banyak situs populer seperti WordPress dibangun dengan PHP.
- **Swift**: Dibuat oleh Apple, bahasa ini digunakan untuk membuat aplikasi di perangkat Apple seperti iPhone, iPad, dan Mac.
- **SQL**: Bukan bahasa untuk membuat aplikasi, tapi khusus untuk berkomunikasi dan mengelola data di dalam database.

Mengenal IDE (Integrated Development Environment)

Apa itu IDE?

IDE adalah sebuah perangkat lunak yang menyediakan semua yang dibutuhkan seorang programmer untuk mengembangkan aplikasi dalam satu tempat.

Bayangkan jika kita ingin melukis, kita butuh kuas, kanvas, dan cat. IDE adalah "studio seni" lengkap yang menyediakan semua alat itu dalam satu paket.

Komponen Utama IDE

- **Editor Kode**: Tempat untuk menulis dan mengedit kode.
- **Compiler/Interpreter**: Alat yang menerjemahkan kode yang kita tulis ke dalam bahasa yang dimengerti komputer. Untuk Python, ini disebut interpreter.
- **Debugger**: Alat untuk mencari dan memperbaiki kesalahan (bug) dalam kode.

Contoh Aplikasi IDE untuk Python

- **PyCharm**: IDE yang sangat populer dan canggih, cocok untuk proyek yang lebih besar. Ada versi gratisnya (Community Edition) yang sudah lebih dari cukup untuk pemula.
- **Visual Studio Code (VS Code)**: Ini adalah editor kode yang sangat ringan dan serbaguna. Dengan menambahkan ekstensi (plugins) untuk Python, VS Code bisa berfungsi layaknya IDE. Ini adalah pilihan yang bagus karena fleksibel dan banyak digunakan.
- **IDLE**: IDE bawaan yang sudah terpasang saat kita menginstal Python. Sangat sederhana dan cocok untuk pemula yang baru memulai.
- **Thonny** adalah sebuah yang dirancang khusus untuk para pemula dalam belajar bahasa pemrograman **Python**.

Keunggulan Thonny untuk Pemula

Sangat Sederhana: Tampilannya bersih dan tidak banyak tombol atau fitur rumit yang bisa membingungkan.

Mudah Dipasang: Saat kita menginstal Thonny, Python sudah disertakan di dalamnya. Jadi, kita tidak perlu repot-repot menginstal Python secara terpisah.

Debugger yang Mudah Digunakan: Thonny memiliki fitur *debugger* (alat untuk mencari kesalahan) yang unik. kita bisa menjalankan kode selangkah demi selangkah dan melihat bagaimana nilai variabel berubah secara visual. Ini sangat membantu untuk memahami alur kerja program.

Visualisasi Variabel: Ada panel khusus yang menampilkan semua variabel yang sedang digunakan dalam program. Ini membuat kita bisa melihat dengan jelas data apa saja yang tersimpan di setiap variabel.

Konsep Dasar

Variabel: Kotak(wadah) tempat menyimpan data. kita bisa memberikan nama pada kotak itu dan mengisinya dengan berbagai jenis data.

Contoh: nama = "Andi"

Tipe Data: Jenis data yang disimpan dalam variabel.

- str (String): Teks. Contoh: "Halo"
- int (Integer): Bilangan bulat. Contoh: 100
- float: Bilangan desimal. Contoh: 3.14
- bool (Boolean): Nilai benar atau salah (True atau False).

Operator: Simbol yang digunakan untuk melakukan operasi pada data.

- Aritmetika: + (tambah), - (kurang), * (kali), / (bagi).
- Perbandingan: == (sama dengan), > (lebih besar), < (lebih kecil).

Fungsi: Kumpulan kode yang menjalankan tugas tertentu dan bisa dipanggil berulang kali.

- Fungsi print(): Menampilkan output ke layar.
- Fungsi input(): Menerima masukan dari pengguna.

Aturan Utama Penamaan Variabel

1. **Hanya Boleh Menggunakan Huruf, Angka, dan Garis Bawah (_)**

Nama variabel tidak boleh mengandung spasi atau karakter khusus lainnya, seperti !, @, #, $, %, dll.

Contoh Benar: nama_lengkap, nilai_siswa1, angka_kedua

Contoh Salah: nama lengkap, nilai-siswa, 1nilai

2. **Tidak Boleh Diawali dengan Angka**

Nama variabel harus selalu diawali dengan huruf (a-z, A-Z) atau garis bawah (_).

Contoh Benar: nama_depan, _hasil

Contoh Salah: 1nama, 2nilai

3. **Bersifat *Case-Sensitive***

Python membedakan antara huruf besar dan huruf kecil. Artinya, nama dan Nama adalah dua variabel yang berbeda.

Contoh:

```
Python
nama = "Budi"

Nama = "Andi"



print(nama)  # Output: Budi

print(Nama)  # Output: Andi
```

4. **Tidak Boleh Menggunakan Kata Kunci (Keywords) Bawaan Python**

Ada beberapa kata yang sudah dipakai oleh Python untuk fungsi tertentu. kita tidak bisa menggunakannya sebagai nama variabel. Contohnya: if, for, while, class, def, True, False, None.

Konvensi yang Baik (Bukan Aturan, Tapi Sangat Dianjurkan)

Selain aturan di atas, ada beberapa konvensi penamaan yang biasa digunakan oleh para programmer Python agar kodenya lebih rapi:

Gunakan Huruf Kecil dengan Garis Bawah (Snake Case)

Untuk nama variabel yang terdiri dari lebih dari satu kata, pisahkan kata-kata tersebut dengan garis bawah (_).

Contoh: umur_siswa, total_harga, nama_depan

Ini adalah konvensi yang paling umum dalam komunitas Python.

Berikan Nama yang Jelas dan Deskriptif

Nama variabel sebaiknya mencerminkan data apa yang disimpan di dalamnya. Hindari nama yang terlalu singkat seperti x, y, atau z jika tidak diperlukan.

Contoh: Daripada x = 50000, lebih baik gaji_pokok = 50000.

Hindari Menggunakan Huruf Kapital Semua

Penamaan dengan huruf kapital semua (contoh: TOTAL_HARGA) biasanya hanya digunakan untuk **konstanta**, yaitu variabel yang nilainya tidak akan pernah berubah.

Mengikuti aturan dan konvensi ini akan membuat kode Pythonmu lebih bersih, mudah dibaca, dan profesional.

Sintaks dasar python

Mencetak Teks dengan *print()*

Ini adalah program "Hello, World!" yang paling dasar.

Python
```
print("Halo, dunia!")
```

Penjelasan: Fungsi *print()* akan menampilkan teks di dalam tanda kurung ke layar.

latihan :

```
--- BIODATA SISWA ---
Nama Lengkap: Daniarsyah
Jenis Kelamin : Perempuan
Umur: 16 tahun
Asal Sekolah: SMA Negeri 6 Cimahi
Hobi: Bermain musik dan membaca buku
Cita-cita: Menjadi seorang insinyur perangkat lunak
---------------------
```

simpan dengan nama file *: biodata_nama_kelas.py*

Menggunakan Variabel

Python
```
nama = "Budi"
umur = 17

print(nama)
print(umur)
```

Python
```
nama = "Budi"
umur = 17

print(“Nama saya : ”, nama)
print(“Umur saya : ”,umur)
```

Apa Itu f-string?

f-string adalah cara yang paling modern, mudah, dan efisien untuk memformat string (teks) di Python. Nama "f-string" adalah singkatan dari **"formatted string literal"**. Dengan f-string, bisa menyisipkan nilai variabel, ekspresi matematika, atau bahkan hasil dari fungsi langsung ke dalam sebuah string dengan cara yang sangat rapi.

Sebelumnya, untuk menggabungkan teks dengan variabel, kita sering menggunakan cara seperti tanda tambah (+) atau metode .format(). Namun, f-string menyederhanakan proses ini.

Cara Penggunaan f-string

Untuk menggunakan f-string, kita hanya perlu meletakkan huruf f di depan tanda kutip (bisa kutip tunggal ' atau ganda "). Di dalam string, kita bisa meletakkan nama variabel atau ekspresi di dalam kurung kurawal {}.

Contoh Sederhana

Bayangkan kita punya data nama dan usia:

Python
```
nama = "Ani"
usia = 16
```

Jika ingin mencetak kalimat "Nama saya Ani dan usia saya 16 tahun.", inilah perbandingannya:

Tanpa f-string (Menggunakan +):

Python
```
print("Nama saya " + nama + " dan usia saya " + str(usia) + " tahun.")
```

Cara ini terlihat rumit karena harus menggunakan + berkali-kali dan mengubah angka menjadi string (str(usia)).

Tanpa f-string (Menggunakan .format()):

Python
print("Nama saya {} dan usia saya {} tahun.".format(nama, usia))

Cara ini lebih baik, tapi masih terlihat kurang intuitif.

Menggunakan f-string:

Python
print(f"Nama saya {nama} dan usia saya {usia} tahun.")

Sangat mudah dibaca dan ringkas! serta bisa melihat langsung variabel mana yang akan disisipkan.

Contoh Lain dengan Ekspresi

dengan f-string bisa juga melakukan perhitungan langsung di dalam kurung kurawal {}:

Python
harga = 50000
jumlah = 2
Menghitung total harga langsung di dalam f-string
*print(f"Total harga untuk {jumlah} barang adalah Rp{harga * jumlah}.")*

Output:

Total harga untuk 2 barang adalah Rp100000.

Mengapa f-string Lebih Disukai?

Mudah Dibaca: Sintaksnya intuitif dan mirip dengan bahasa natural.

Lebih Cepat: Secara performa, f-string lebih cepat daripada metode pemformatan string lainnya.

Fleksibel: Bisa menyisipkan variabel, ekspresi, dan bahkan memanggil fungsi di dalamnya.

Program Python untuk Mencetak Biodata

Kode di bawah ini akan menyimpan data biodata ke dalam variabel, lalu mencetaknya ke layar dengan rapi.

Python
```
# Program Biodata Sederhana
# Menyimpan informasi biodata ke dalam variabel

nama = "Daniarsyah"
umur = 16
```

```
asal_sekolah = "SMA Negeri 6 Cimahi"
hobi = "Bermain musik dan membaca buku"
cita_cita = "Menjadi seorang insinyur perangkat lunak"
```

Penjelasan: Setiap baris di atas adalah contoh dari **variabel** yang menyimpan data. Misalnya, variabel nama menyimpan nilai "Daniarsyah", dan variabel umur menyimpan angka 16.

Mencetak Informasi Biodata

Selanjutnya, kita akan menggunakan fungsi print() untuk menampilkan semua data tersebut. Kita bisa menggunakan **f-string** agar tampilannya lebih rapi.

Python
```
# Mencetak semua informasi dengan format yang rapi
print("--- BIODATA SISWA ---")
print(f"Nama Lengkap: {nama}")
print(f"Umur: {umur} tahun")
print(f"Asal Sekolah: {asal_sekolah}")
print(f"Hobi: {hobi}")
print(f"Cita-cita: {cita_cita}")
print("----------------------")
```

Penjelasan:

print("--- BIODATA SISWA ---") akan mencetak teks biasa.

print(f"Nama Lengkap: {nama}") adalah contoh f-string. Teks di dalam tanda kutip akan dicetak, dan nilai dari variabel nama akan disisipkan di dalam kurung kurawal {}.

Jika seluruh kode di atas digabungkan dan dijalankan, outputnya akan terlihat seperti ini:

```
--- BIODATA SISWA ---
Nama Lengkap: Daniarsyah
Umur: 16 tahun
Asal Sekolah: SMA Negeri 6 Cimahi
Hobi: Bermain musik dan membaca buku
Cita-cita: Menjadi seorang insinyur perangkat lunak
----------------------
```

Dengan cara ini, kita bisa membuat program yang menyimpan dan menampilkan informasi dengan mudah.

Mengenal Fungsi `input()`

Dalam membuat program, terkadang kita perlu mendapatkan informasi dari pengguna. Misalnya, saat membuat program kalkulator, kita perlu menanyakan angka apa yang ingin dihitung. Nah, di Python, kita menggunakan fungsi **input()** untuk hal ini.

Fungsi input() berfungsi untuk:

Menampilkan sebuah pesan (opsional) kepada pengguna.

Menunggu pengguna mengetikkan sesuatu di *keyboard*.

Membaca dan menyimpan apa yang diketik pengguna.

Data yang dimasukkan oleh pengguna akan selalu disimpan dalam bentuk **string (str)**, bahkan jika yang diketik adalah angka. Ini adalah poin penting yang harus kita ingat!

Sintaks Dasar

Sintaks atau cara penulisan fungsi input() adalah sebagai berikut:

Python
```
nama_variabel = input("Pesan yang akan ditampilkan: ")
```

nama_variabel: Variabel ini akan menyimpan data yang dimasukkan oleh pengguna.

"Pesan yang akan ditampilkan: ": Ini adalah teks petunjuk yang akan muncul di layar, memberitahu pengguna apa yang harus mereka masukkan. Teks ini sifatnya opsional, tapi sangat disarankan agar program menjadi lebih jelas.

Contoh Penggunaan Sederhana

Mari kita coba membuat program sederhana untuk menanyakan nama.

Python
```
# Program untuk menanyakan nama
nama_pengguna = input("Masukkan nama Anda: ")

print(f"Halo, {nama_pengguna}! Selamat datang di program ini.")
```

Penjelasan:

Saat program dijalankan, teks Masukkan nama Anda: akan muncul di layar.

Program akan berhenti sejenak, menunggu kita mengetikkan sesuatu dan menekan Enter.

Apa pun yang kita ketik, misalnya Budi, akan disimpan ke dalam variabel nama_pengguna.

Baris print() akan menampilkan pesan yang sudah digabungkan dengan nama yang kita masukkan.

Mengubah Tipe Data dari `input()`

Karena input() selalu menghasilkan **string**, kita perlu mengubahnya (konversi) jika ingin menggunakan data tersebut untuk perhitungan matematika. Kita bisa menggunakan fungsi **`int()`** untuk mengubahnya menjadi bilangan bulat (*integer*) atau **`float()`** untuk bilangan desimal.

Contoh Konversi Tipe Data

Bayangkan kita ingin membuat program untuk menghitung usia.

Python
```
# Program untuk menghitung usia
tahun_sekarang = 2024
# input() akan menghasilkan string, jadi harus dikonversi
tahun_lahir_str = input("Masukkan tahun lahir Anda: ")
tahun_lahir_int = int(tahun_lahir_str) # Konversi string ke integer

usia = tahun_sekarang - tahun_lahir_int

print(f"Tahun ini Anda berusia {usia} tahun.")
```

Penjelasan:

input() akan membaca tahun lahir sebagai teks (contoh: "2007").

int(tahun_lahir_str) akan mengubah teks "2007" menjadi angka 2007.

Tanpa konversi ini, Python tidak bisa melakukan operasi pengurangan (-), dan akan muncul *error*.

kita bisa menyederhanakan kode di atas dengan langsung melakukan konversi:

Python
```
tahun_sekarang = 2024
tahun_lahir = int(input("Masukkan tahun lahir Anda: "))
usia = tahun_sekarang - tahun_lahir
print(f"Tahun ini Anda berusia {usia} tahun.")
```

Dengan menguasai fungsi input(), kita bisa mulai membuat program yang interaktif dan dinamis!