AVARAGE CALCULATOR

Disusun Oleh :

Sandy Tsalsa Fanany (202210370311171)

Hilma Salman Maulana (202210370311174)

T. M. Dhabit Tarim (202210370311177)

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH MALANG

TAHUN PELAJARAN 2023/2024

DAFTAR ISI

BAB I

Informal Spesification

Kode yang diberikan adalah program Python yang membuat GUI sederhana untuk menghitung rata-rata dari daftar angka. Menggunakan library tkinter untuk membuat antarmuka grafis dengan label, widget entry untuk input angka, dan tombol untuk menghitung rata-rata. Ketika tombol ditekan, program mendapatkan input angka, menghitung rata-rata, dan menampilkan hasil dengan dua angka decimal dalam kotak pesan. Kode mengatur kesalahan potensial, seperti input yang tidak valid, dengan menampilkan pesan kesalahan dalam kotak pesan.

Informasi informal tentang spesifikasi kode adalah sebagai berikut:

- Program membuat jendela GUI bernama "Average Calculator" menggunakan library tkinter.
- Antarmuka mencakup label yang mengundangkan pengguna untuk memasukkan angka terpisah dengan tanda koma dan widget entry untuk input angka.
- Ketika tombol "Hitung Rata-rata" (Menghitung Rata-rata) ditekan, program mendapatkan input angka, menghitung rata-rata, dan menampilkan hasil dengan dua angka decimal dalam kotak pesan.
- Jika input tidak valid (misalnya, bukan list angka), kode menampilkan pesan kesalahan dalam kotak pesan.
- Program menggunakan blok try-except untuk mengatasi kemungkinan ValueError saat memproses input.

BAB II

Formal Program Spesification dengan Hoare Triple

Hoare Triple adalah notasi matematis yang digunakan untuk spesifikasi formal dari sebuah program. Notasi ini terdiri dari tiga bagian: sebuah kondisi pra (precondition), sebuah pernyataan program, dan sebuah kondisi pasca (postcondition). Kondisi pra adalah kondisi yang harus terpenuhi sebelum eksekusi program, sedangkan kondisi pasca adalah kondisi yang diharapkan terpenuhi setelah eksekusi program.

{Input: numbers_str adalah string yang berisi angka-angka yang dipisahkan oleh tanda koma}

{Output: average adalah rata-rata dari angka-angka yang diinput}

1. {numbers_str adalah string yang berisi angka-angka yang dipisahkan oleh tanda koma}
2. {numbers adalah list dari angka-angka yang diinput}
3. numbers_str = entry_numbers.get()
4. numbers = [float(num) for num in numbers_str.split(",")]
5. {numbers berisi setidaknya satu angka}
6. {average adalah rata-rata dari angka-angka yang diinput}
7. average = sum(numbers) / len(numbers)
8. {average berisi rata-rata dari angka-angka yang diinput}
9. {Jika input tidak valid, tampilkan pesan kesalahan}
10. if not numbers:
11. raise ValueError("Please enter valid numbers.")
12. {Pesan kesalahan ditampilkan}
13. {Jika input valid, tampilkan hasil rata-rata}
14. messagebox.showinfo("Result", f"Rata-ratanya adalah = {average:.2f}")
15. {Hasil rata-rata ditampilkan}

BAB III

Error Correctness

1. Error Detection, Program tersebut mencakup mekanisme untuk mendeteksi kesalahan, terutama dalam hal input. Penggunaan blok try-except memungkinkan program untuk mendeteksi kesalahan saat mengonversi input string menjadi bilangan desimal. Hal ini merupakan langkah yang baik dalam memastikan error detection.

2. Error Handling, Program juga menyertakan penanganan kesalahan dengan menampilkan pesan kesalahan jika input tidak valid. Ini penting untuk memberikan umpan balik kepada pengguna dan mencegah kegagalan program akibat kesalahan input.

3. Data Integrity, Dalam konteks error correctness, penting untuk memastikan integritas data. Program tersebut melakukan perhitungan rata-rata dan menampilkan hasilnya. Dalam hal ini, perlu dipastikan bahwa data yang diolah sesuai dengan harapan dan tidak terjadi distorsi atau kesalahan dalam perhitungan.

4. Robustness, Keberhasilan program dalam mengatasi kesalahan input dan memastikan perhitungan yang benar merupakan indikator dari tingkat robustness atau ketahanan program terhadap kesalahan.

5. valueError, Kesalahan ini terjadi ketika pengguna memasukkan input yang tidak valid, seperti string kosong atau string yang tidak dapat diubah menjadi angka. Program menangani kesalahan ini dengan menampilkan pesan kesalahan dalam kotak pesan menggunakan messagebox.showerror().

6. TypeError, Kesalahan ini terjadi ketika program mencoba melakukan operasi yang tidak valid pada tipe data yang salah. Dalam program ini, kesalahan ini mungkin terjadi jika numbers bukan tipe data list. Namun, karena numbers selalu diinisialisasi sebagai list, kesalahan ini tidak mungkin terjadi.

7. ZeroDivisionError: Kesalahan ini terjadi ketika program mencoba membagi angka dengan nol. Dalam program ini, kesalahan ini mungkin terjadi jika pengguna memasukkan daftar angka kosong. Namun, program menangani kesalahan ini dengan menampilkan pesan kesalahan dalam kotak pesan menggunakan messagebox.showerror().

BAB IV

Code Review

1. Validasi Input: Kode ini melakukan validasi input untuk memastikan bahwa daftar angka yang dimasukkan oleh pengguna valid sebelum menghitung rata-ratanya. Namun, validasi tambahan dapat ditambahkan untuk menangani kasus-kasus khusus, seperti spasi yang tidak disengaja sebelum atau sesudah tanda koma.
2. Penanganan Kesalahan: Kode ini menangani kesalahan yang mungkin terjadi saat menghitung rata-rata dari daftar angka dengan menampilkan pesan kesalahan yang sesuai dalam kotak pesan. Namun, penanganan kesalahan tambahan dapat ditambahkan untuk menangani kasus-kasus khusus, seperti kesalahan saat membagi angka dengan nol.
3. Komentar: Kode ini dapat ditingkatkan dengan penambahan komentar yang menjelaskan bagaimana setiap bagian dari program bekerja, terutama untuk orang lain yang mungkin membaca atau mengembangkan kode ini di masa depan.
4. Pemisahan Fungsi: Kode ini memisahkan fungsionalitas perhitungan rata-rata ke dalam sebuah fungsi ‘calculate_average()’. Ini membuat kode lebih terstruktur dan mudah untuk dikelola.

BAB VI

Defensive programming

Defensive programming adalah pendekatan pengembangan perangkat lunak yang bertujuan untuk menghindari kesalahan dan meminimalkan dampak kesalahan yang terjadi. Berikut adalah beberapa saran untuk menerapkan defensive programming pada kode yang diberikan:

1. Validasi Input: Kode ini melakukan validasi input untuk memastikan bahwa daftar angka yang dimasukkan oleh pengguna valid sebelum menghitung rata-ratanya. Namun, validasi tambahan dapat ditambahkan untuk menangani kasus-kasus khusus, seperti spasi yang tidak disengaja sebelum atau sesudah tanda koma. Selain itu, penggunaan **try-except** dapat digunakan untuk menangani kesalahan yang mungkin terjadi saat mengonversi input menjadi angka.

2. Penanganan Kesalahan: Kode ini menangani kesalahan yang mungkin terjadi saat menghitung rata-rata dari daftar angka dengan menampilkan pesan kesalahan yang sesuai dalam kotak pesan. Namun, penanganan kesalahan tambahan dapat ditambahkan untuk menangani kasus-kasus khusus, seperti kesalahan saat membagi angka dengan nol.

3. Pemisahan Fungsi: Kode ini memisahkan fungsionalitas perhitungan rata-rata ke dalam sebuah fungsi **calculate_average()**. Ini membuat kode lebih terstruktur dan mudah untuk dikelola. Namun, pemisahan fungsi tambahan dapat dilakukan untuk memisahkan fungsionalitas yang berbeda dan memudahkan pengembangan dan pemeliharaan kode di masa depan.

4. Komentar: Kode ini dapat ditingkatkan dengan penambahan komentar yang menjelaskan bagaimana setiap bagian dari program bekerja, terutama untuk orang lain yang mungkin membaca atau mengembangkan kode ini di masa depan. Komentar dapat membantu memperjelas tujuan dan fungsionalitas kode, serta membantu pengembang lain memahami kode dengan lebih baik.

5. Unit Testing: Unit testing dapat digunakan untuk memastikan bahwa kode berfungsi dengan benar dan menghindari kesalahan yang mungkin terjadi. Unit testing dapat dilakukan dengan membuat skrip pengujian yang menguji fungsionalitas kode dalam berbagai situasi dan kondisi.

Dengan menerapkan defensive programming pada kode yang diberikan, kita dapat meminimalkan kesalahan dan memastikan bahwa program berjalan dengan benar dalam berbagai situasi dan kondisi.

BAB VI

Testing

Unit testing adalah proses pengujian perangkat lunak di mana setiap unit atau komponen individual diuji untuk memastikan bahwa mereka bekerja dengan benar. Unit testing dilakukan untuk memastikan bahwa setiap unit kode software sudah bisa bekerja sesuai harapan. Dalam konteks kode yang diberikan, unit testing dilakukan untuk memeriksa fungsi **calculate_average()** apakah berjalan sesuai harapan atau tidak. Beberapa prinsip ilmiah yang dapat diterapkan dalam unit testing antara lain:

1. Reproduktif: Pengujian harus dapat direproduksi, artinya hasil pengujian harus konsisten ketika dijalankan berkali-kali. Hal ini memastikan bahwa pengujian dapat diandalkan dan hasilnya dapat dipercaya.
2. Independen: Setiap pengujian harus independen satu sama lain, artinya hasil dari satu pengujian tidak boleh memengaruhi hasil pengujian lainnya. Hal ini memastikan bahwa setiap kasus uji dapat dijalankan secara terpisah dan tidak saling bergantung.
3. Komprehensif: Pengujian harus mencakup semua kemungkinan skenario yang mungkin terjadi, baik skenario normal maupun skenario ekstrem. Hal ini memastikan bahwa kode diuji secara komprehensif dan dapat menangani berbagai kondisi.
4. Dokumentasi: Hasil dari pengujian harus didokumentasikan dengan baik, termasuk skenario pengujian, hasil yang diharapkan, dan hasil aktual. Hal ini memastikan bahwa setiap perubahan atau pembaruan kode dapat dipantau dan diverifikasi.

Adapun pengujian yang dilakukan kami untuk menguji berjalnnya code kami untuk mendapatkan hasil yang sesuai, Pengujian ini mencakup tiga kasus:

1. test_calculate_average_valid_input(): Uji fungsi **calculate_average()** dengan input valid, yaitu list angka yang benar. Pengujian ini mengevaluasi apakah fungsi bekerja sesuai harapan dengan membandingkan hasil yang diberikan dengan nilai harapan (3.0).

2. test_calculate_average_empty_input(): Uji fungsi **calculate_average()** dengan input kosong (list angka kosong). Pengujian ini mengevaluasi apakah fungsi mengangkat **ValueError** ketika diberikan input yang tidak valid.
3. test_calculate_average_invalid_input(): Uji fungsi **calculate_average()** dengan input yang tidak valid, yaitu list angka yang mencakup elemen non-angka. Pengujian ini mengevaluasi apakah fungsi mengangkat **ValueError** ketika diberikan input yang tidak valid.

Pengujian ini digunakan untuk memastikan bahwa fungsi **calculate_average()** bekerja dengan benar dan mengorbankan kasus ketidakpastian.

BAV VII

Versioning Control

Versioning control, juga dikenal sebagai kontrol versi, adalah praktik melacak dan mengelola perubahan pada kode perangkat lunak. Sistem kontrol versi membantu tim pengembang perangkat lunak dalam mengelola perubahan pada kode sumber. Dengan menggunakan sistem kontrol versi, tim pengembang dapat melacak setiap modifikasi pada kode, membandingkan versi kode yang berbeda, dan mengembalikan kode ke versi sebelumnya jika diperlukan.Terdapat beberapa manfaat utama dari penggunaan sistem kontrol versi, antara lain:

- Melacak perubahan: Sistem kontrol versi memungkinkan tim pengembang untuk melacak setiap perubahan yang dilakukan pada kode, termasuk siapa yang melakukan perubahan dan kapan perubahan tersebut dilakukan.
- Kolaborasi: Dengan menggunakan sistem kontrol versi, tim pengembang dapat bekerja secara kolaboratif pada proyek perangkat lunak, tanpa takut akan konflik perubahan yang dilakukan oleh anggota tim yang berbeda.
- Pemulihan: Sistem kontrol versi memungkinkan tim pengembang untuk mengembalikan kode ke versi sebelumnya jika diperlukan, misalnya jika terjadi kesalahan yang tidak diinginkan pada kode.

Dalam konteks kode yang diberikan, penggunaan sistem kontrol versi sangat penting untuk memastikan manajemen perubahan yang efektif pada kode perangkat lunak. Dengan menggunakan sistem kontrol versi, tim pengembang dapat memastikan bahwa setiap perubahan pada kode terdokumentasi dengan baik, dan dapat dikelola dengan efisien. Berikut adalah tahapan untuk melakukan Versioning Control di Git Hub:

1. Buat akun GitHub: Langkah pertama adalah membuat akun GitHub jika Anda belum memiliki satu. Kunjungi situs web GitHub dan ikuti instruksi untuk membuat akun.
2. Buat repositori: Setelah membuat akun, buat repositori baru untuk proyek Anda. Repositori adalah tempat di mana kode sumber Anda akan disimpan dan dikelola. Untuk membuat repositori baru, klik tombol "New" di halaman utama GitHub dan ikuti instruksi untuk membuat repositori baru.

3. Inisialisasi Git: Setelah membuat repositori, inisialisasi Git pada proyek Anda. Git adalah sistem kontrol versi yang akan digunakan untuk melacak perubahan pada kode sumber Anda. Untuk menginisialisasi Git, buka terminal atau command prompt dan arahkan ke direktori proyek Anda. Kemudian ketik perintah **git init** untuk menginisialisasi Git.
4. Tambahkan file ke repositori: Setelah menginisialisasi Git, tambahkan file ke repositori Anda. Untuk menambahkan file, ketik perintah **git add <nama file>** di terminal atau command prompt. Anda juga dapat menambahkan semua file dengan mengetik perintah **git add .**.
5. Buat commit: Setelah menambahkan file, buat commit untuk menyimpan perubahan pada kode sumber Anda. Commit adalah tindakan untuk menyimpan perubahan pada kode sumber Anda ke repositori. Untuk membuat commit, ketik perintah **git commit -m "pesan commit"** di terminal atau command prompt.
6. Push ke GitHub: Setelah membuat commit, push perubahan ke repositori di GitHub. Untuk melakukan push, ketik perintah **git push origin master** di terminal atau command prompt. Perintah ini akan mengirim perubahan pada kode sumber Anda ke repositori di GitHub.
7. Lakukan perubahan: Setiap kali Anda melakukan perubahan pada kode sumber Anda, ulangi langkah 4 hingga 6 untuk menambahkan, membuat commit, dan push perubahan ke repositori di GitHub.

BAB VIII

Implentasi API

1. Pemahaman Konsep API

Sebelum Anda mulai mengimplementasikan API, penting untuk memahami konsep dasar API. Berikut adalah beberapa poin penting yang perlu diperhatikan:

- API adalah suatu tautan yang menyatakan perilaku dan fitur dari suatu program untuk program lain.
- API memungkinkan berbagai pengguna untuk menggunakan fitur yang ditawarkan oleh suatu program tanpa memahami cara pengiriman internalnya.
- API harus menyediakan dokumentasi yang jelas dan tepat untuk memahami pengguna yang akan menggunakannya.

2. Mengimplementasikan API dalam Kode

Berikut adalah langkah-langkah untuk mengimplementasikan API dalam kode yang Anda berikan:

1. Identifikasi fitur dan perilaku yang akan Anda implementasikan sebagai API.
2. Buat dokumentasi yang jelas dan tepat untuk setiap fitur dan perilaku yang akan Anda implementasikan sebagai API.
3. Gunakan nama yang unik dan jelas untuk mengidentifikasi setiap fitur dan perilaku yang akan Anda implementasikan sebagai API.
4. Pastikan bahwa setiap fitur dan perilaku yang Anda implementasikan sebagai API bekerja dengan baik dan tidak menyebabkan kesalahan.
5. Lakukan tes kasus (unit testing) untuk memastikan bahwa setiap fitur dan perilaku yang Anda implementasikan sebagai API bekerja dengan baik.

3. Pengujian dan Pemantauan API

Setelah Anda mengimplementasikan API, penting untuk melakukan pengujian dan pemantauan untuk memastikan bahwa API Anda bekerja dengan baik dan tidak menyebabkan kesalahan. Berikut adalah beberapa metode yang dapat Anda gunakan untuk melakukan pengujian dan pemantauan:

- Menggunakan tes kasus (unit testing) untuk memastikan bahwa setiap fitur dan perilaku yang Anda implementasikan sebagai API bekerja dengan baik.
- Menggunakan alat pemantauan API untuk memantau penggunaan API dan mengidentifikasi masalah jika terjadi.
- Mengumpulkan dokumentasi yang jelas dan tepat untuk membantu pengguna memahami cara menggunakan API Anda.

Dengan mengikuti langkah-langkah di atas, Anda akan dapat mengimplementasikan API yang efisien dan aman dalam kode Anda.

BAB IX

GUI

Dalam kode yang diberikan, GUI dibuat menggunakan modul **tkinter** yang merupakan modul standar untuk membuat antarmuka pengguna pada Python. GUI tersebut terdiri dari beberapa elemen antarmuka pengguna, seperti label, entry, dan button.

- Label: Label digunakan untuk menampilkan teks yang menjelaskan fungsi dari elemen antarmuka pengguna lainnya. Pada kode tersebut, label digunakan untuk menampilkan teks "Masukkan Angka (yang dipisahkan oleh tanda koma):".
- Entry: Entry digunakan untuk memungkinkan pengguna memasukkan input, dalam hal ini angka-angka yang akan dihitung rata-ratanya. Pada kode tersebut, entry digunakan untuk memungkinkan pengguna memasukkan angka-angka yang akan dihitung rata-ratanya.
- Button: Button digunakan untuk memicu fungsi **calculate_average()** ketika pengguna menekan tombol "Hitung Rata-rata". Pada kode tersebut, button digunakan untuk memicu fungsi **calculate_average()** ketika pengguna menekan tombol "Hitung Rata-rata".

Fungsi **calculate_average()** akan mengambil input dari elemen entry, memproses input tersebut, dan menampilkan hasilnya menggunakan modul **messagebox**. Jika input yang dimasukkan tidak valid, seperti tidak ada angka yang dimasukkan, maka fungsi akan mengangkat **ValueError** dan menampilkan pesan kesalahan menggunakan modul **messagebox**.Dalam kode tersebut, GUI dibuat dengan sederhana dan mudah dipahami. Namun, pengembang dapat menambahkan elemen antarmuka pengguna lainnya, seperti menu, gambar, dan grafik, untuk meningkatkan fungsionalitas dan kegunaan dari program.