ASSERTION

DOSEN PENGAMPU:

FANNI SUKMA, S.ST,. M.T

NOVI, S.Kom., M.T

DISUSUN OLEH:

KELOMPOK 4

1. DENI RAMADHAN	**(2211083010)**
2. FADILA ISLAMI NISA	**(2211082007)**
3. PUTI HANIFAH MARSLA	**(2211082024)**
4. NURUL AULIA	**(2211082023)**
5. WINALDO AGENG	**(2111082047)**

PRODI TEKNOLOGI REKAYASA PERANGKAT LUNAK

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI PADANG

2024

ASSERTION

A. Penjelasan Assertion

Assertion adalah pernyataan yang digunakan dalam kode untuk memeriksa apakah kondisi tertentu benar. Jika kondisi tersebut salah (False), program akan berhenti dan menghasilkan AssertionError. Pernyataan asersi sangat berguna selama pengembangan dan debugging untuk memastikan bahwa program berjalan sesuai dengan asumsi yang diinginkan.

B. Tujuan dan Manfaat Assertion

1. Deteksi Kesalahan:

   Assertion membantu mendeteksi kesalahan logika dalam kode dengan memastikan bahwa kondisi tertentu harus selalu benar di titik tertentu dalam program.

2. Dokumentasi Kode:

   Assertion berfungsi sebagai dokumentasi langsung dalam kode, menunjukkan asumsi yang dibuat oleh programmer pada titik tertentu.

3. Debugging:

   Assertion memudahkan debugging dengan menunjukkan dengan jelas di mana asumsi dalam kode tidak berlaku.

4. Keandalan Kode:

   Dengan memastikan kondisi penting selalu benar, assertion meningkatkan keandalan dan stabilitas program.

C. Cara Kerja Assertion

Assertion bekerja dengan memeriksa kondisi boolean. Jika kondisi tersebut bernilai true, eksekusi program berlanjut seperti biasa. Jika kondisi bernilai false, assertion biasanya akan memicu kesalahan (error) atau pengecualian (exception), dan eksekusi program dapat dihentikan atau dialihkan.

D. Syntax Assertion

```
assert condition, message
```

penjelasan:

- `condition` adalah ekspresi yang diharapkan benar.
- `message` adalah pesan opsional yang akan ditampilkan jika `condition` adalah `False`.

E. Contoh Penggunaan Assertion Menggunakan Bahasa Python

1. Validasi Input dalam Fungsi Bagi

   Misalnya kita memiliki fungsi untuk membagi dua angka. Kita ingin memastikan bahwa pembagi (denominator) tidak boleh nol.

```python
#Validasi input dalam fungsi bagi
def bagi(x, y):
    assert y != 0, "Penyebut tidak boleh nol"
    return x / y

# Penggunaan:
print(bagi(10, 2))  # Output: 5.0
print(bagi(10, 0))  # AssertionError: Penyebut tidak boleh nol

```

Setelah itu Run file

```
PS C:\Users\LENOVO> & C:/Users/LENOVO/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/Validasi Input.py"
5.0
Traceback (most recent call last):
  File "d:\Semester 4\Validasi Input.py", line 8, in <module>
    print(bagi(10, 0))  # AssertionError: Penyebut tidak boleh nol
          ^^^^^^^^^^^
  File "d:\Semester 4\Validasi Input.py", line 3, in bagi
    assert y != 0, "Penyebut tidak boleh nol"
AssertionError: Penyebut tidak boleh nol
```

2. Memastikan angka positif

   Kita ingin memastikan bahwa angka yang di inputkan adalah positif (>0) tidak boleh negatif.

```python
#Memastikan angka positif
def check_positive(number):
    assert number > 0, "Angka harus positif"
    return f"Angka {number} adalah positif"

print(check_positive(-3)) # AssertionError: angka tidak boleh negatif
print(check_positive(5))

```

Setelah melakukan Run File

```
PS C:\Users\LENOVO> & C:/Users/LENOVO/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/
"
Traceback (most recent call last):
  File "d:\Semester 4\Evolusi dan Konstruksi PL\Angka poditif.py", line 6, in <module>
    print(check_positive(-3)) # AssertionError: angka tidak boleh negatif
          ^^^^^^^^^^^^^^^^^^
  File "d:\Semester 4\Evolusi dan Konstruksi PL\Angka poditif.py", line 3, in check_positive
    assert number > 0, "Angka harus positif"
AssertionError: Angka harus positif
PS C:\Users\LENOVO>
```

3. Memastikan usia diatas 18 tahun

```python
def cek_usia(usia):
    assert usia > 18, "Usia harus lebih dari 18 tahun"
    print(f"Usia {usia} tahun diterima")

def main():
    usia = 20
    cek_usia(usia)

    usia_lain = 17
    cek_usia(usia_lain)  # AssertionError: Usia kurang dari 17

if __name__ == "__main__":
    main()
```

Setelah Melakukan RunFile:

```
PS C:\Users\LENOVO> & C:/Users/LENOVO/AppData/Local/Programs/Python/Python311/python.exe "d:/Sem
y"
Usia 20 tahun diterima
Traceback (most recent call last):
  File "d:\Semester 4\Evolusi dan Konstruksi PL\Usia diatas 18.py", line 13, in <module>
    main()
  File "d:\Semester 4\Evolusi dan Konstruksi PL\Usia diatas 18.py", line 10, in main
    cek_usia(usia_lain)  # AssertionError: Usia kurang dari 17
    ^^^^^^^^^^^^^^^^^^^
  File "d:\Semester 4\Evolusi dan Konstruksi PL\Usia diatas 18.py", line 2, in cek_usia
    assert usia > 18, "Usia harus lebih dari 18 tahun"
AssertionError: Usia harus lebih dari 18 tahun
```

4. Memastikan tipe data String

```python
def cek_string(my_var):
    assert isinstance(my_var, str), "Variabel harus bertipe string"
    print(f"'{my_var}' adalah string")

def main():
    my_var = "Hello, world!"
    cek_string(my_var)

    my_var_lain = 12345
    cek_string(my_var_lain)  # AssertionError: Variabel tidak bertipe string

if __name__ == "__main__":
    main()
```

Setelah melakukan RunFile:

```
PS C:\Users\LENOVO> & C:/Users/LENOVO/AppData/Local/Programs/Python/Python311/python.exe "d
'Hello, world!' adalah string
Traceback (most recent call last):
  File "d:\Semester 4\Evolusi dan Konstruksi PL\Cek string.py", line 13, in <module>
    main()
  File "d:\Semester 4\Evolusi dan Konstruksi PL\Cek string.py", line 10, in main
    cek_string(my_var_lain)  # AssertionError: Variabel tidak bertipe string
    ^^^^^^^^^^^^^^^^^^^^^^^
  File "d:\Semester 4\Evolusi dan Konstruksi PL\Cek string.py", line 2, in cek_string
    assert isinstance(my_var, str), "Variabel harus bertipe string"
AssertionError: Variabel harus bertipe string
```

5. Memastikan Panjang daftar list

```python
def cek_panjang_daftar(my_list, panjang_minimum):
    assert len(my_list) >= panjang_minimum, f"Daftar harus memiliki setidaknya {panjang_minimum} elemen"
    print(f"Daftar memiliki {len(my_list)} elemen, memenuhi syarat panjang minimum {panjang_minimum}")

def main():
    daftar = [1, 2, 3, 4, 5]
    panjang_minimum = 3
    cek_panjang_daftar(daftar, panjang_minimum)

    daftar_pendek = [1, 2]
    cek_panjang_daftar(daftar_pendek, panjang_minimum)  # Ini akan membangkitkan AssertionError

if __name__ == "__main__":
    main()
```

Setelah melakukan RunFile:

```
PS C:\Users\LENOVO> & C:/Users/LENOVO/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/Evolus
ar.py"
Daftar memiliki 5 elemen, memenuhi syarat panjang minimum 3
Traceback (most recent call last):
  File "d:\Semester 4\Evolusi dan Konstruksi PL\cek panjang daftar.py", line 14, in <module>
    main()
  File "d:\Semester 4\Evolusi dan Konstruksi PL\cek panjang daftar.py", line 11, in main
    cek_panjang_daftar(daftar_pendek, panjang_minimum)  # Ini akan membangkitkan AssertionError
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "d:\Semester 4\Evolusi dan Konstruksi PL\cek panjang daftar.py", line 2, in cek_panjang_daftar
    assert len(my_list) >= panjang_minimum, f"Daftar harus memiliki setidaknya {panjang_minimum} elemen"
AssertionError: Daftar harus memiliki setidaknya 3 elemen
```

6. Memastikan Tidak Ada nilai kosong dalam daftar

```python
def cek_tidak_ada_nilai_kosong(my_list):
    assert all(my_list), "Daftar tidak boleh mengandung nilai kosong"
    print(f"Daftar tidak mengandung nilai kosong: {my_list}")

def main():
    daftar = [1, 2, 3, 4, 5]
    cek_tidak_ada_nilai_kosong(daftar)

    daftar_kosong = [1, 2, 0, 4, 5]
    cek_tidak_ada_nilai_kosong(daftar_kosong)  # Ini akan membangkitkan AssertionError

if __name__ == "__main__":
    main()
```

Setelah melakukan RunFile:

```
PS C:\Users\LENOVO> & C:/Users/LENOVO/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/Evolusi dan Konstr
lai kosong.py"
Daftar tidak mengandung nilai kosong: [1, 2, 3, 4, 5]
Traceback (most recent call last):
  File "d:\Semester 4\Evolusi dan Konstruksi PL\cek tidak ada nilai kosong.py", line 13, in <module>
    main()
  File "d:\Semester 4\Evolusi dan Konstruksi PL\cek tidak ada nilai kosong.py", line 10, in main
    cek_tidak_ada_nilai_kosong(daftar_kosong)  # Ini akan membangkitkan AssertionError
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "d:\Semester 4\Evolusi dan Konstruksi PL\cek tidak ada nilai kosong.py", line 2, in cek_tidak_ada_nilai_kosong
    assert all(my_list), "Daftar tidak boleh mengandung nilai kosong"
AssertionError: Daftar tidak boleh mengandung nilai kosong
```

F. Kesimpulan

Dalam pengembangan perangkat lunak, assertion merupakan alat yang penting untuk memastikan keandalan dan kualitas kode. Dengan menggunakan assertion, kita dapat secara eksplisit memeriksa asumsi-asumsi yang dibuat dalam kode, sehingga memungkinkan kita untuk mendeteksi kesalahan atau bug secara dini. Assertion juga membantu dalam meningkatkan keamanan kode dengan memverifikasi bahwa kondisi-kondisi penting terpenuhi. Pesan kesalahan yang dihasilkan oleh assertion memberikan petunjuk yang jelas tentang sumber masalah, sehingga memudahkan dalam proses debugging.