PENERAPAN DECORATOR TERHADAP VALIDASI MASUKAN DATA BERUPA BILANGAN PRIMA DALAM SEBUAH LIST

Deva Anjani Khayyuninafsyah (122450014), Nabiilah Putri Karnaia (122450029),
Chevando Daffa Pramanda (122450095), Novelia Adinda (122450104),
Rafly Prabu Darmawan (122450140)

Program Studi Sains Data, Fakultas Sains, Institut Teknologi Sumatera
Jl. Terusan Ryacudu, Way Huwi, Kec. Jatiagung, Kabupaten Lampung Selatan, Lampung 35365

Email:
deva.122450014@student.itera.ac.id, nabiilah.122450029@student.itera.ac.id,
chevando.122450095@student.itera.ac.id, novelia.122450104@student.itera.ac.id,
rafly.122450140@student.itera.ac.id

1. Pendahuluan

Pembuatan program komputer mendasari berbagai aspek, termasuk ketika kita berurusan dengan bilangan prima. Namun, agar program-program tersebut dapat berfungsi dengan baik dan aman, hal yang krusial adalah memastikan bahwa data yang dimasukkan ke dalamnya valid dan sesuai dengan yang diharapkan. *Decorator* sendiri merupakan sebuah alat dalam pemrograman yang muncul sebagai solusi yang sangat berguna dalam hal ini. Dengan menggunakan fungsi *Decorator*, kita bisa melakukan pemeriksaan otomatis terhadap data sebelum program dijalankan. Hal ini membantu memastikan bahwa data yang dimasukkan sesuai dengan kebutuhan program, terutama dalam kasus penghitungan bilangan prima yang membutuhkan input yang benar.

Dalam artikel ini, kita akan memahami konsep dan implementasi dari fungsi *Decorator* untuk validasi data dalam konteks program bilangan prima. Kita akan membahas bagaimana fungsi *decorator* dapat mempermudah proses validasi input, sehingga meningkatkan keamanan dan kehandalan program secara keseluruhan. Melalui penelusuran ini, kita juga akan mempertimbangkan pentingnya validasi data dalam pengembangan perangkat lunak. Li (2018) .

Diharapkan dengan pemahaman yang lebih mendalam tentang fungsi *decorator* untuk validasi data, pembaca akan dapat mengimplementasikannya secara efektif dalam pengembangan program-program yang berkaitan dengan bilangan prima.

2. Metode

a. Decorator

Dalam pemrograman *Python*, *Decorator* adalah fungsi yang dapat mengubah fungsi atau metode lain dengan menambahkan fungsionalitas tambahan ke dalamnya

tanpa mengubah kode aslinya secara signifikan. *Decorator* biasanya digunakan untuk menambahkan perilaku seperti *logging, caching*, atau validasi ke fungsi yang sudah ada. Ramalho, Luciano. (2015).

b. Validasi Data

Validasi data adalah proses verifikasi dan penilaian data yang dikumpulkan atau dibuat untuk memastikan bahwa mereka dapat diandalkan dan sesuai dengan standar yang ditetapkan. Tujuan utama dari validasi data adalah untuk memastikan bahwa data yang digunakan dalam analisis atau pemrosesan lebih lanjut dapat diandalkan dan konsisten. Proses validasi data dapat mencakup berbagai langkah, seperti memeriksa format data dan menemukan nilai yang tidak valid atau hilang. Hair, J.F., Black, W.C., Babin, B.J., Anderson, R.E., & Tatham, R.L. (2019).

c. Fungsi Bilangan Prima

Algoritma sederhana biasanya digunakan di *Python* untuk menjalankan fungsi untuk menentukan apakah suatu bilangan adalah bilangan prima. Algoritma ini menentukan apakah bilangan tersebut hanya dapat dibagi oleh 1 dan dirinya sendiri. Langtangen, H. P. (2014).

d. Validate Input

Dalam Python, validasi input adalah proses yang memeriksa apakah input yang diberikan kepada suatu fungsi atau program memenuhi format atau persyaratan yang diinginkan sebelum melanjutkan pemrosesan. Ini membantu mencegah kesalahan atau kegagalan yang disebabkan oleh data yang tidak valid. Ramalho, L. (2015).

3. Pembahasan

```python
def bilangan_prima(n):
  if n <= 1:
    return False
  if n <= 3:
    return True
  if n % 2 == 0 or n % 3 == 0:
    return False
  i = 5
  while i * i <= n:
    if n % i == 0 or n % (i + 2) == 0:
      return False
    i += 6
  return True
```

Gambar 1

Deret kode pada gambar pertama merupakan sebuah fungsi awal sederhana, yaitu membuat fungsi "bilangan_prima". Fungsi tersebut menerima satu argumen yaitu "n" yang merupakan sebuah bilangan yang ingin ditentukan. Lalu fungsi diberikan beberapa kondisi, yang pertama yaitu "n" akan diperiksa apakah kurang dari atau sama dengan 1, jika benar maka output mencetak "*return False*" artinya bukan bilangan prima. Kemudian yang kedua, jika “n” diperiksa kurang dari atau sama dengan 3, maka output akan

mencetak "*return True*" artinya merupakan bilangan prima. Selanjutnya yang ketiga, jika "n" diperiksa akan habis dibagi 2 atau 3 maka output akan mencetak "*return False*" artinya bukan bilangan prima.

Pada fungsi "bilangan_prima" juga terdapat looping "*while*" yang dimulai dari iterasi "i=5". *Looping* akan berhenti ketika "i * i <= n" untuk memastikan bahwa i tidak melebihi bilangan "n" yang diperiksa. Selanjutnya diberikan kondisi jika bilangan "n" habis dibagi "i" atau "i+2" akan mengembalikan nilai "*False*" artinya "n" bukan bilangan prima. Jika tidak terjadi pembagian yang menyebabkan "n" bukan bilangan prima, maka "n" adalah bilangan prima dan fungsi akan mengembalikan nilai "*True*".

```
def validate_input(func):
  def wrapper(*args, **kwargs):
    bukan_prima = []
    for arg in args:
      if isinstance(arg, list):
        for num in arg:
          if not bilangan_prima(num):
            bukan_prima.append(num)
        if bukan_prima:
          print("Error: Terdapat elemen bukan bilangan prima:")
          for num in bukan_prima:
            print(num, end=" ")
          return None
        else:
          return func(*args, **kwargs)
      else:
        print("Error: Input harus berupa list.")
        return None
  return wrapper
@validate_input
def cek_prima(nums):
  print("Semua elemen dalam list merupakan bilangan prima.")
```

Gambar 2

Pada deret kode pada gambar kedua, decorator akan didefinisikan untuk memvalidasi input dengan fungsi validate_input dan fungsi wrapper. Fungsi wrapper merupakan fungsi di dalam fungsi yang akan dipanggil sebagai argumen decorator. Setelah melakukan validasi input, kode akan melakukan iterasi melalui semua argumen.

a. Jika argumen adalah *list* membuat iterasi melalui tiap elemen yang ada pada *list* dan Jika elemen pada *list* bukan bilangan prima kita akan menambahkan elemen yang bukan bilangan prima kedalam *list* bukan_prima.
b. Jika ada elemen yang bukan bilangan prima dalam *list* akan mencetak pesan *error* yang kemudian akan menghentikan fungsi dan mengembalikannya ke *None*.
c. Jika semua elemen dalam *list* adalah bilangan prima, maka akan dilanjutkan dengan menjalankan fungsi *Decorator*.

Fungsi validate_input merupakan sebuah decorator yang akan menvalidasi input fungsi cek_prima. Fungsi ini akan mengambil argumen dalam bentuk list dan akan memeriksa tiap elemen didalamnya apakah bilangan prima atau bukan. Fungsi cek_prima yang memiliki argumen nums adalah fungsi utama dengan penggunaan decorator yang

akan mencetak pesan jika semua elemen dalam list argumen nums merupakan bilangan prima.

```
# Input disimpan dalam variabel listKu sebagai list
listKu = [2, 3, 5, 7, 11, 13]
# Panggil fungsi cek_prima dengan variabel input listKu
cek_prima(listKu)
```

```
Semua elemen dalam list merupakan bilangan prima.
```

Gambar 3

Pada deret kode di gambar ketiga ini merupakan penggunaan dari pembuatan fungsi sebelumnya. Pada kode diatas terdapat pemanggilan fungsi "cek_prima(listKu)" dimana inputan "listKu" merupakan sebuah *list* yang berisi beberapa bilangan. *Output* yang dihasilkan dari pemanggilan fungsi tersebut adalah "Semua elemen dalam *list* merupakan bilangan prima", menunjukan bahwa semua elemen dalam *list* tersebut bilangan prima.

```
# Input disimpan dalam variabel listKu sebagai list
listKu = [1, 2, 3, 4, 5, 6]
# Panggil fungsi cek_prima dengan variabel input listKu
cek_prima(listKu)
```

```
Error: Terdapat elemen bukan bilangan prima:
1 4 6
```

Gambar 4

Deret kode pada gambar keempat ini memiliki kode yang hampir sama seperti pada Gambar 3, dimana sedikit perbedaan pada gambar ini yaitu elemen dalam *list* "listKu" terdapat bilangan yang bukan bilangan prima. Maka *output* yang dicetak adalah "Error: Terdapat elemen bukan bilangan prima: 1 4 6" dimana bilangan dalam *list* yaitu 1, 4, 6 bukan termasuk bilangan prima.

```
# Input disimpan dalam variabel listKu sebagai list
listKu = (2, 3, 4, 5, 7, 8)
# Panggil fungsi cek_prima dengan variabel input listKu
cek_prima(listKu)
```

```
Error: Input harus berupa list.
```

Gambar 5

Pada deret kode gambar kelima ini terdapat kesalahan saat pemanggilan fungsi "cek_prima(listKu)". Dimana *input* "listKu" pada gambar sebenarnya adalah *tuple* bukan sebuah *list*. Oleh karena itu, fungsi *decorator* *@validate_input* akan memeriksa bahwa *input* bukan sebuah *list* dan akan mencetak pesan "*Error*: Input harus berupa *list*".

4. Kesimpulan

Saat memvalidasi suatu data kita dapat menggunakan fungsi *decorator* yang merupakan bilangan prima dalam sebuah list. *Decorator* sendiri adalah sebuah fungsi yang dapat mengubah fungsi atau metode lain dengan menambahkan fungsionalitas tambahan ke dalamnya tanpa mengubah kode aslinya secara signifikan. Dalam artikel ini digunakan 4 jenis metode yaitu *decorator*, validasi data dan fungsi bilangan prima.

Dalam pembahasan kode diatas memiliki 2 fungsi utama yaitu bilangan_prima(n) dan *validate_input*(*func*) dan *decorator @validate_input* yang digunakan untuk dapat memastikan bahwa semua elemen dalam *list* merupakan bilangan prima sebelum menjalankan fungsi cek_prima. Kode tersebut mengimplementasikan dua fungsi yang saling berhubungan. Fungsi bilangan_prima akan memeriksa apakah suatu bilangan adalah merupakan bilangan prima, sementara fungsi *validate_input* berfungsi sebagai *decorator* yang memastikan *input* yang diberikan adalah *list* yang mengandung bilangan-bilangan prima sebelum menjalankan fungsi cek_prima. Dengan ini, kode tersebut memastikan bahwa setiap elemen dalam *list* yang diberikan pada fungsi cek_prima merupakan bilangan prima sebelum melakukan proses lebih lanjut.

5. Daftar Pustaka

Li, X., & Ding, Z. (2018). Decorator-Based Model for High-Level Programming Education. International Journal of Emerging Technologies in Learning (iJET), 13(2), 96-108.

Ramalho, Luciano. (2015). Fluent Python, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Hair, J.F., Black, W.C., Babin, B.J., Anderson, R.E., & Tatham, R.L. (2019). Multivariate Data Analysis. Cengage Learning.

Langtangen, H. P. (2014). A Primer on Scientific Programming with Python. Springer

Sweigart, A. (2019). Automate the Boring Stuff with Python. No Starch Press, Inc.

Ramalho, L. (2015). Fluent Python: Clear, Concise, and Effective Programming. O'Reilly Media.