**LAPORAN TUGAS KECIL 4**

IF2251 Strategi Algoritmik

Ekstraksi Informasi dari Artikel Berita dengan Algoritma Pencocokan String

Semester II Tahun 2019/2020



Valentinus Devin Setiadi - 13518116 - K2

**PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**2020**

**BAB 1**

**Deskripsi Persoalan**

Algoritma pencocokan string (pattern) Knuth-Morris-Pratt (KMP) dan Algoritma Boyer-Moore merupakan algoritma yang lebih baik daripada brute force. Pada Tugas Kecil IV kali ini Anda diminta membuat aplikasi sederhana ekstraksi informasi dengan kedua algoritma tersebut, plus menggunakan regular expression (regex). Teks yang akan Anda proses adalah teks berita berbahasa Indonesia seperti contoh berikut ini (jabar11042020.txt).

|  |
| --- |
| 421 Orang di Jabar Terkonfirmasi Positif COVID-19Yudha Maulana - detikNewsSabtu, 11 Apr 2020 20:07 WIB **Bandung** - Angka positif virus [Corona atau COVID-19](https://www.detik.com/tag/virus-corona?tag_from=ridwan-kamil) di Jawa Barat menembus angka 400 kasus. Laman Pusat Informasi dan Koordinasi COVID-19 Jabar (Pikobar) pada Sabtu (11/4/2020) pukul 18.43 WIB, mencatat terdapat 421 orang yang terkonfirmasi positif COVID-19.  Dibandingkan sehari sebelumnya, jumlah tercatat yaitu 388 orang. Terjadi penambahan 8,5 persen atau 33 kasus per harinya. Sementara itu, secara nasional terdapat 3.842 kasus positif COVID-19.  Dari 421 kasus tersebut, 40 orang meninggal dunia dengan keterangan terpapar [COVID-19](https://www.detik.com/tag/virus-corona?tag_from=ridwan-kamil). Sedangkan, angka kesembuhan di Jabar masih tetap berada di angka 19 orang.  Per hari jumlah Orang Dalam Pemantauan (ODP) di Jabar mencapai 28.775 orang. Sebanyak 15.363 di antaranya masih menjalani proses pemantauan dan 13.412 orang lainnya telah selesai menjalani proses pemantauan.  Sementara itu jumlah Pasien Dalam Pengawasan (PDP) mencapai 2.278 orang. Tercatat 1.344 orang masih menjalani proses pengawasan dan 934 orang lainnya telah selesai menjalani proses pengawasan. |

Pada kumpulan teks berita korban covid-19 ini, informasi penting dari pengguna adalah jumlah korban dan waktunya. Oleh karena itu, informasi yang akan diekstraksi adalah angka (diberi warna biru) dan waktu (diberi warna merah).

Pengguna aplikasi ini akan memberikan masukan berupa folder yang berisi kumpulan teks berita, keywords, dan hasil ekstraksi jumlah dan waktunya. Karena sebagian besar kalimat mengandung angka, aplikasi akan memfilter angka berdasarkan keywords dari pengguna, seperti ‘terkonfirmasi positif’, ‘meninggal dunia’, ‘Orang Dalam Pemantauan’, ‘ODP’, ‘Pasien Dalam Pengawasan’, ‘PDP’ atau keyword lainnya. Hasilnya berupa pasangan angka dan waktu, serta kalimat yang mengandung informasi tersebut. Waktu yang diambil harus berada dalam satu kalimat dengan angka tersebut. Jika tidak ada, gunakan tanggal artikel yang tercantum. Jika terdapat lebih dari satu angka, pilih angka yang paling dekat dengan keyword. Berikut contohnya.

Keyword: *terkonfirmasi positif*

Hasil ekstraksi informasi:

Jumlah: *421;* Waktu: Sabtu, 11 Apr 2020 20:07 WIB

# 421 Orang di Jabar **Terkonfirmasi Positif** COVID-19. (jabar11042020.txt)

Jumlah: *421;* Waktu: *Sabtu (11/4/2020) pukul 18.43 WIB*

Laman Pusat Informasi dan Koordinasi COVID-19 Jabar (Pikobar) pada Sabtu (11/4/2020) pukul 18.43 WIB, mencatat terdapat 421 orang yang **terkonfirmasi positif** COVID-19. (jabar11042020.txt)

Keyword: *meninggal dunia*

Hasil ekstraksi informasi:

Jumlah: *40;* Waktu: Sabtu, 11 Apr 2020 20:07 WIB

Dari 421 kasus tersebut, 40 orang **meninggal dunia** dengan keterangan terpapar [COVID-19](https://www.detik.com/tag/virus-corona?tag_from=ridwan-kamil). (jabar11042020.txt)

Terdapat dua jenis pencocokan string yang Anda lakukan. Pertama, exact match dengan keyword yang diberikan pengguna untuk memfilter kalimat yang akan diproses informasinya. Semua teknik (KMP, BM, dan regex) bisa digunakan untuk fitur ini. Kedua, ekstraksi jumlah dan waktu dari kalimat hasil exact match dengan menggunakan regex.

Pencarian tidak bersifat *case sensitive*, jadi huruf besar dan huruf kecil dianggap sama (hal ini dapat dilakukan dengan mengganggap seluruh karakter di dalam pattern dan teks sebagai huruf kecil semua atau huruf kapital semua).

**BAB 2**

**Landasan Teori**

* **KMP** (Knuth-Morris-Pratt)

Algoritma Knuth-Morris-Pratt (KMP) mencari pola dalam teks dalam urutan kiri ke kanan (seperti algoritma brute force). Tetapi terdapat perbedaan yang membuat algoritma ini lebih efektif yaitu dengan menggeser polanya tidak seperti algoritma brute force. Apabila terjadi mismatch pada teks dengan pola yang dicari, algoritma akan mengambil informasi tertentu untuk menentukan dimanakah selanjutnya pencarian dilakukan, dengan demikian pengecekan dapat dilakukan secara tidak berurutan tetapi tetap akan melewati pengecekan string yang sudah dicek sebelumnya, di program ini saya melakukannya dengan membentuk sebuah array yang berisi nilai dari prefix suffix terbesar.

* **Boyer-Moore**

Algoritma dari Boyer-Moore terdiri dari 2 teknik mendasar yaitu *The Looking Glass Technique* dengan mencari pattern pada teks dalam urutan kanan ke kiri / *backwards* dan  *The character-jump technique* yang apabila terjadi *mismatch* pattern dapat melewati beberapa karakter setelahnya dengan syarat-syarat tertentu.

* **Regex**

Regex atau regular expression adalah ekspresi regular dalam Bahasa pemrograman python yang merupakan suatu metode pencocokan pattern. Pada pencocokan string dengan regex, kita dapat memanfaatkan library regex yang telah ada seperti re.findall() atau re.search(). Regex dalam program ini berguna juga dalam pencarian angka dan tanggal dalam text.

**BAB 2**

**Kode Program**

**Boyer-Moore**

|  |
| --- |
| **def arr(pattern): #array temp**  **arr\_bm = {}**  **for i in range(len(pattern)):**  **arr\_bm[pattern[i]] = i**  **return arr\_bm**  **def boym(pattern, text): #mencari pattern dalam teks dengan algoritma boyer moore**  **pat = pattern.lower()**  **txt = text.lower()**    **m = len(pat)**  **n = len(txt)**  **arr\_bm = arr(pat)**  **arr\_match = []**  **# s is shift of the pattern**  **s = 0**  **while(s <= n-m):**  **j = m-1**  **while j>=0 and txt[s+j] == pat[j]:**  **j -= 1**  **if j<0:**  **arr\_match.append(s)**  **if s+m<n:**  **if (txt[s+m] in arr\_bm):**  **s += m-arr\_bm[txt[s+m]]**  **else:**  **s += m+1**  **else:**  **s += 1**  **else:**  **if (txt[s+j] in arr\_bm):**  **s += max(1, j-arr\_bm[txt[s+j]])**  **else:**  **s += max(1, j+1)**  **return len(arr\_match)>0** |

**Knuth-Morris-Pratt**

|  |
| --- |
| **def kmpArr(pat, m, x):#array dengan prefix dan suffix terpanjang**  **pattern = pat.lower()**  **len = 0 # length of the previous longest prefix suffix**  **i = 1**  **while i < m:**  **if pattern[i] == pattern[len]:**  **len += 1**  **x[i] = len**  **i += 1**  **else: #pattern[i]!= pattern[len]:**  **if len != 0:**  **len = x[len-1]**  **else: #len==0**  **x[i] = 0**  **i += 1**  **return x**    **def kmp(pat, txt):#mencari pattern pada text menggunakan algoritma kmp**  **pattern = pat.lower()**  **text = txt.lower()**  **n = len(text)**  **m = len(pattern)**  **x =[0 for y in range (m)] #arr of the longest length of prefix and suffix**  **arr = kmpArr(pattern,m,x)**  **i = 0**  **j = 0**  **arr\_match =[]**  **while i<n:**  **if text[i] == pattern[j]:**  **i+=1**  **j+=1**    **if j == m :**  **arr\_match.append(i-j)**  **j = arr[j-1]**    **elif i < n and pattern [j] != text[i]:**  **if j != 0:**  **j = arr[j-1]**  **else: #j==0**  **i+=1**  **return(len(arr\_match)>0)**  **def kmp\_txt(arr):**  **hasil="match at index"**  **for i in arr:**  **hasil+=" "+str(i)**  **return hasil** |

**Regex**

|  |
| --- |
| **import re**  **from nltk.tokenize import sent\_tokenize**  **from kmp import \***  **from bm import \***  **def splitArr(teks):#memecah 1teks(paragraf) menjadi beberapa kalimat dengan memanfaatkan nltk**  **splt = sent\_tokenize(teks)**  **return splt**  **def closestDig(kalimat,p):#method mencari potongan kalimat yang terdapat angka dan pattern dari user**  **pattern = p.lower()**  **sentence = kalimat.lower()**  **pat = '[0-9]'**  **result = re.search(genPattern(True,pat, pattern),sentence)**  **if result is None:**  **return ("no result")**  **return result.group()**  **def getDNumber(matchClosest):#method mencari angka dari potongan kalimat**  **pat =r' [0-9]\*\.[0-9]+| [0-9]+'**  **space = " "**  **matchClosest = space + matchClosest**  **result = re.search(pat,matchClosest)**  **if result is None:**  **return ("no result")**  **return result.group()**  **def findDate(sentence):#method mencari waktu dengan format hari tanggal waktu waktubagian**  **regex = r"((Senin|Monday|Selasa|Tuesday|Rabu|Wednesday|Kamis|Thursday|Jumat|Friday|Sabtu|Saturday|Minggu|Sunday)(?:.)\*?((\d{1,2}(\/|\-|\.)\d{1,2}(\/|\-|\.)\d{4})|(\d{1,2}( )(Jan|Januari|January|Feb|Februari|February|Mar|Maret|March|Apr|April|Mei|May|Jun|Juni|June|Jul|July|Juli|Aug|Agustus|August|Sep|September|Oct|Oktober|October|November|Nov|Dec|December|Desember)( )\d{4}))(?:.)\*?(([0-1]\d|[2][0-3])((\:|\.)[0-5]\d){1,2})((?:.)\*?WIB(?:.)\*?|(?:.)\*?WITA(?:.)\*?| (?:.)\*?WIT(?:.)\*?|))"**  **result = re.search(regex, sentence)**  **if result is None:**  **date = r"((\d{1,2}(\/|\-|\.)\d{1,2}(\/|\-|\.)\d{4})|(\d{1,2}( )(Jan|Januari|January|Feb|Februari|February|Mar|Maret|March|Apr|April|Mei|May|Jun|Juni|June|Jul|July|Juli|Aug|Agustus|August|Sep|September|Oct|Oktober|October|November|Nov|Dec|December|Desember)( )\d{4}))"**  **result = re.search (date,sentence)**  **if result is None:**  **return ("no date")**  **return result.group()**  **def searchRe(pattern,text):#mencari pattern dalam teks dengan regex**  **pat = pattern.lower()**  **txt = text.lower()**  **reg = re.findall("{}".format(pat),txt)**  **return (len(reg)>0)**  **def genPattern(isMultiple, pattern, lookup):**  **sym = ""**  **if isMultiple:**  **sym = "+"**  **return r"({pattern}{sym}(?:(?!( {pattern})).)\*?{lookup}|{lookup}(?:(?! ({pattern}) ).)\*?( {pattern}\*\.{pattern}+| {pattern}{sym}))".format(pattern = pattern, sym = sym, lookup = lookup)** |

**main.py**

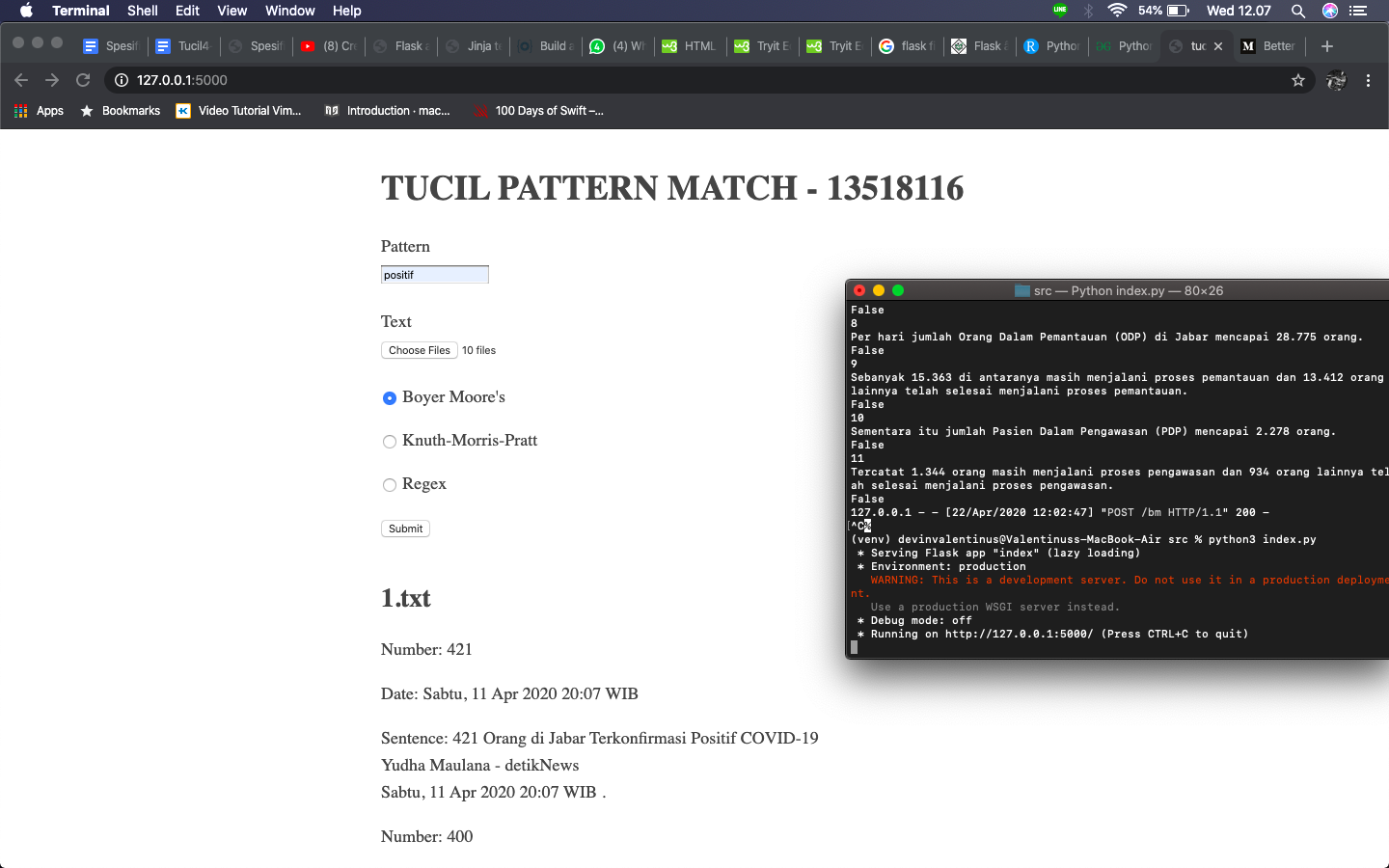
|  |
| --- |
| **import re**  **from kmp import \***  **from bm import \***  **from regx import \***  **'''**  **method algoritma boyer moore mengembalikan dictionary hasil yaitu tanggal angka dan kalimat**  **'''**  **def readBm(naskah,pattern):**  **allSen = splitArr(naskah)**  **arrTot =[]**  **dateTemp = findDate(allSen[0])**  **print(arrTot)**  **print(len(arrTot))**  **for j in range (len(allSen)):**  **arrtemp =[]**  **dict={}**  **print(j)**  **print(allSen[j])**  **date =findDate(allSen[j])**  **valid = boym(pattern,allSen[j])**  **print(valid)**  **if valid:**  **sentClose = closestDig(allSen[j],pattern)**  **digit = getDNumber(sentClose)**  **print ("ini digit nya"+str(digit))**  **dict["angka"] = digit**  **dict["kalimat"] =allSen[j]**  **if (date =="no date"):**  **dict["tanggal"] = dateTemp**  **else:**  **dict["tanggal"] = date**  **arrTot.append(dict)**  **return (arrTot)**  **'''**  **method algoritma Knuth-Morris-Pratt mengembalikan dictionary hasil yaitu tanggal angka dan kalimat**  **'''**  **def readKMP(naskah,pattern):**  **allSen = splitArr(naskah)**  **arrTot =[]**  **dateTemp = findDate(allSen[0])**  **print(arrTot)**  **print(len(arrTot))**  **for j in range (len(allSen)):**  **arrtemp = []**  **dict={}**  **print(j)**  **print(allSen[j])**  **date =findDate(allSen[j])**  **valid = kmp(pattern,allSen[j])**  **print(valid)**  **if valid:**  **sentClose = closestDig(allSen[j],pattern)**  **digit = getDNumber(sentClose)**  **print ("ini digit nya"+str(digit))**  **dict["angka"] = digit**  **dict["kalimat"] =allSen[j]**  **if (date =="no date"):**  **dict["tanggal"] = dateTemp**  **else:**  **dict["tanggal"] = date**  **arrTot.append(dict)**  **return (arrTot)**  **'''**  **method algoritma regex mengembalikan dictionary hasil yaitu tanggal angka dan kalimat**  **'''**  **def readRegx(naskah,pattern):**  **allSen = splitArr(naskah)**  **arrTot =[]**  **dateTemp = findDate(allSen[0])**  **print(arrTot)**  **print(len(arrTot))**  **for j in range (len(allSen)):**  **arrtemp=[]**  **dict={}**  **date =findDate(allSen[j])**  **valid = searchRe(pattern,allSen[j])**  **if valid:**  **sentClose = closestDig(allSen[j],pattern)**  **digit = getDNumber(sentClose)**  **print ("ini digit nya"+str(digit))**  **dict["angka"] = digit**  **dict["kalimat"] =allSen[j]**  **if (date =="no date"):**  **dict["tanggal"] = dateTemp**  **else:**  **dict["tanggal"] = date**  **arrTot.append(dict)**  **return (arrTot)** |

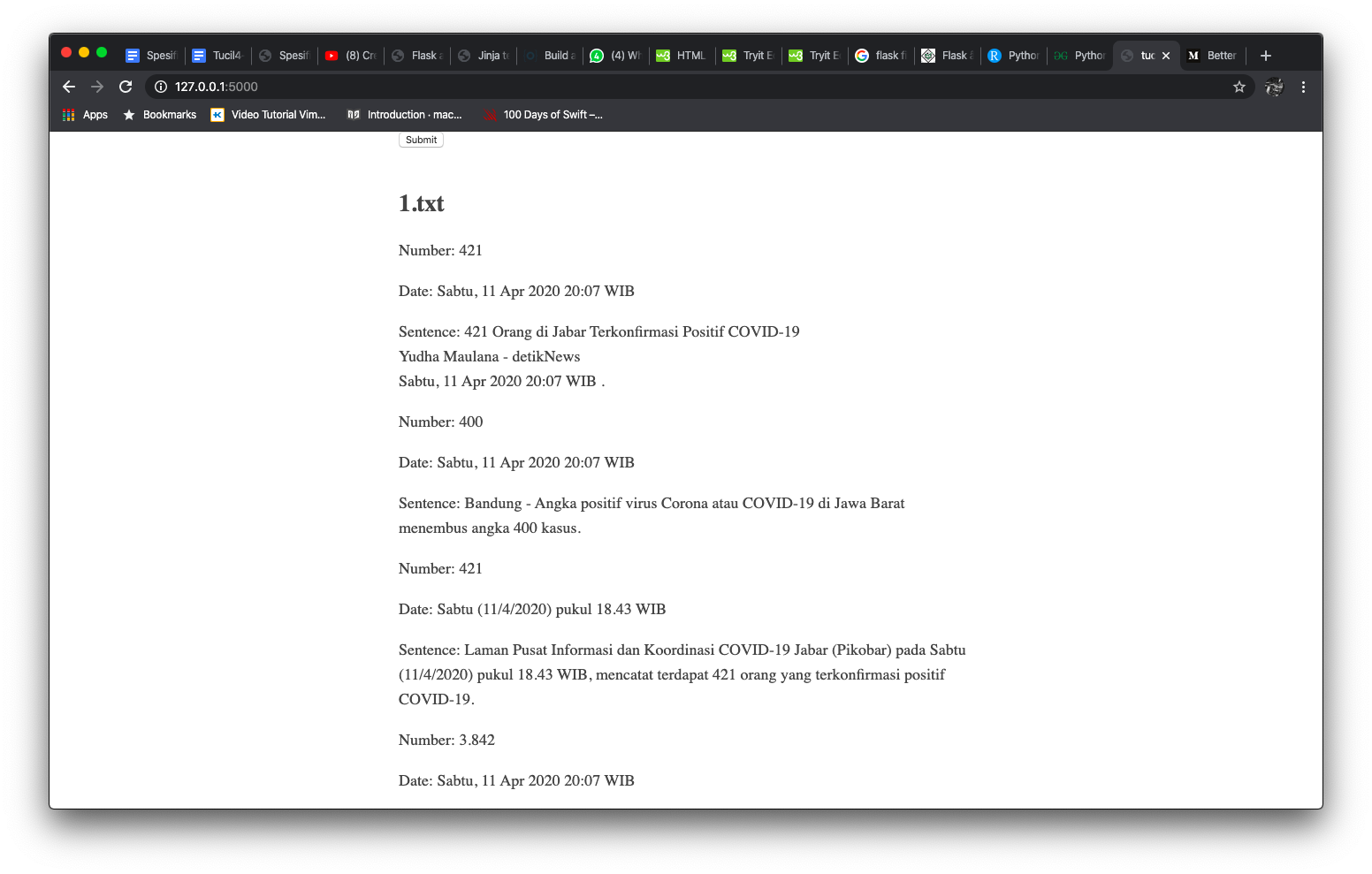
**index.py**

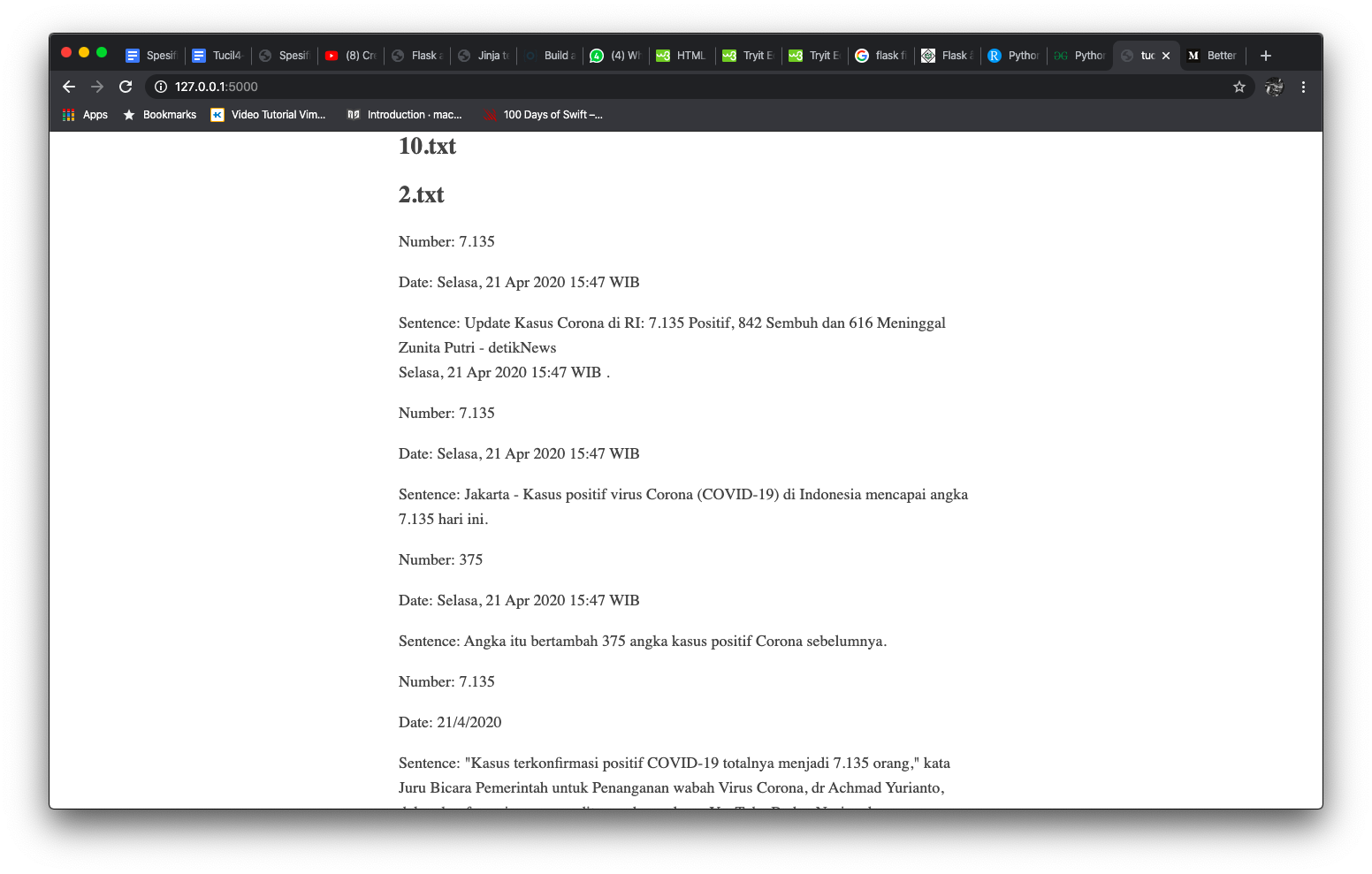
|  |
| --- |
| **from flask import Flask,request,render\_template**  **from kmp import \***  **from bm import \***  **from regx import \***  **from main import \***  **import re**  **app = Flask(\_\_name\_\_)**  **@app.route("/bm", methods=['POST'])**  **def showBm():**  **naskahList = request.files.getlist('naskah')**  **pattern = request.form['pattern']**  **result = {}**  **for item in naskahList:**  **naskah = item.read().decode()**  **result[item.filename] = readBm(naskah,pattern)**  **return result**  **@app.route("/KMP",methods=['POST'])**  **def showKMP():**  **naskahList = request.files.getlist('naskah')**  **pattern = request.form['pattern']**  **result = {}**  **for item in naskahList:**  **naskah = item.read().decode()**  **result[item.filename] = readKMP(naskah,pattern)**  **return result**  **@app.route("/re",methods=['POST'])**  **def showRegx():**  **naskahList = request.files.getlist('naskah')**  **pattern = request.form['pattern']**  **result = {}**  **for item in naskahList:**  **naskah = item.read().decode()**  **result[item.filename] = readRegx(naskah,pattern)**  **return result**  **@app.route("/about",methods=['POST'])**  **def hewwo():**  **return "nim : 13518116"**  **@app.route("/",methods =['GET'])**  **def index():**  **return render\_template("index.html")**  **if \_\_name\_\_ == "\_\_main\_\_":**  **app.run()** |

**BAB 3**

**Screenshot Program**

****

****

****

**BAB 4**

**Spesifikasi Komputer**

****

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi | V |  |
| 2. Program berhasil running | V |  |
| 3. Program dapat menerima input dan menuliskan output. | V |  |
| 4. Luaran sudah benar untuk data uji | V |  |