



Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	71230997
Nama Lengkap	Christ Jevicto Ajimas Kirana
Minggu ke / Materi	10 / Tipe Data Dictionary

SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.

SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN DUTA WACANA
YOGYAKARTA
2024

BAGIAN 1: MATERI MINGGU INI (40%)

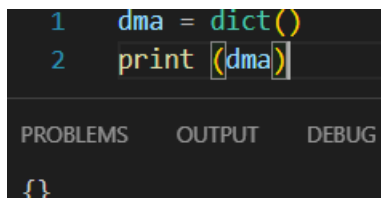
Pada bagian ini, tuliskan kembali semua materi yang telah anda pelajari minggu ini. Sesuaikan penjelasan anda dengan urutan materi yang telah diberikan di saat praktikum. Penjelasan anda harus dilengkapi dengan contoh, gambar/ilustrasi, contoh program (source code) dan outputnya. Idealnya sekitar 5-6 halaman.

MATERI 1

1.1 Materi

Dictionary adalah struktur data dalam Python yang mirip dengan list, tetapi lebih umum karena indeksnya bisa berupa apa pun. Dictionary terdiri dari pasangan **kunci:nilai**, di mana kunci harus unik. Ini digunakan untuk memetakan kunci ke nilai, seperti sebuah kamus yang memetakan kata dalam bahasa Inggris ke bahasa Spanyol. Fungsi `dict()` digunakan untuk membuat dictionary baru yang kosong. Contoh =

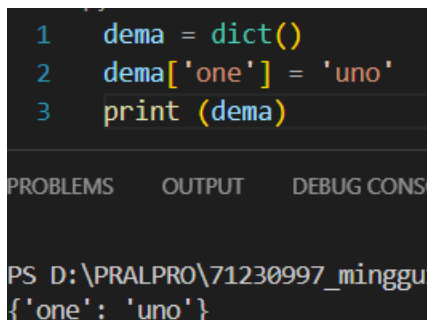
```
1 dma = dict()
2 print (dma)
```



Tanda kurung kurawal `{}` digunakan untuk merepresentasikan dictionary kosong. Untuk menambahkan item dalam dictionary, dapat menggunakan kurung kotak `[]`.

Contoh =

```
1 dema = dict()
2 dema['one'] = 'uno'
3 print (dema)
```



Potongan kode tersebut membuat item yang memetakan kunci 'satu' ke nilai 'uno'. Jika kita mencetak dictionary tersebut, akan terlihat pasangan nilai kunci antara kunci dan nilai tersebut dalam format input. Sebagai contoh, jika kita membuat dictionary baru dengan tiga item dan mencetak keseluruhan datanya, maka format outputnya akan sama dengan format inputnya.

Contoh =

```
>>> eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres'}
>>> print(eng2sp)
{'one': 'uno', 'three': 'tres', 'two': 'dos'}
```

Pengurutan pasangan kunci-nilai dalam sebuah dictionary tidaklah tetap. Meskipun demikian, hal ini tidak menjadi masalah karena akses ke nilai dalam dictionary tidak bergantung pada indeks integer, melainkan menggunakan kunci untuk mencari nilai yang sesuai.

```
1 eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres'}
2 print(eng2sp['two'])
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER COM

PS D:\PRALPRO\71230997_minggu11> & C:/Users/chris/AppData/Local/Programs/Python/Python38-64/Python.exe -i

Contoh =

Urutan item dalam dictionary tidak begitu penting karena setiap kunci memiliki pasangan nilai yang tetap. Jika mencoba mengakses nilai dengan kunci yang tidak ada, akan muncul pengecualian.

```
aid.py > ...
1 eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres'}
2 print(eng2sp['four'])
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER COMM

PS D:\PRALPRO\71230997_minggu11> & C:/Users/chris/AppData/Local/Programs/Python/Python38-64/Python.exe -i

Traceback (most recent call last):
 File "d:\PRALPRO\71230997_minggu11\aid.py", line 2, in <module>
 print(eng2sp['four'])
 ~~~~~^~~~~~  
KeyError: 'four'

Contoh =

Fungsi len pada dictionary digunakan untuk mengembalikan jumlah pasangan nilai kunci. Contoh =

```
aid.py > ...
1 eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres'}
2 print(len(eng2sp))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER COMM

PS D:\PRALPRO\71230997\_minggu11> & C:/Users/chris/AppData/Local/Programs/Python/Python38-64/Python.exe -i

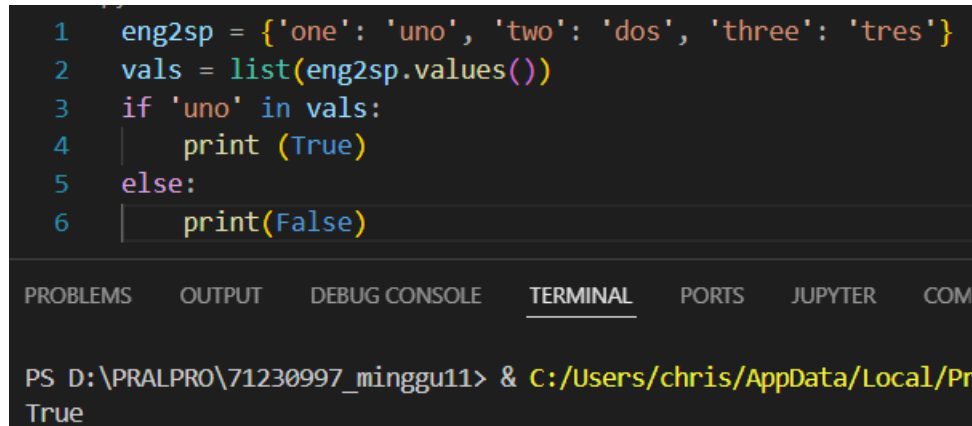
3

Operator in dalam dictionary mengembalikan nilai benar (true) atau salah (false) sesuai dengan kunci yang ada dalam dictionary.

```
>>> 'one' in eng2sp
True
>>> 'uno' in eng2sp
False
```

Contoh =

Gunakan metode `values()` pada dictionary untuk mengambil nilai yang ada. Nilai-nilai ini kemudian dapat dikonversi menjadi daftar dan digunakan dalam operator `in` untuk memeriksa keanggotaan.



```
1 eng2sp = {'one': 'uno', 'two': 'dos', 'three': 'tres'}
2 vals = list(eng2sp.values())
3 if 'uno' in vals:
4     print (True)
5 else:
6     print(False)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER COM

PS D:\PRALPRO\71230997\_minggu11> & C:/Users/chris/AppData/Local/Pr  
True

Contoh =

Pada list, operator `in` menggunakan algoritma pencarian linear, yang berarti waktu pencarian meningkat seiring dengan pertambahan panjang list. Dalam dictionary, Python menggunakan algoritma Hash Table yang memiliki properti yang efisien. Oleh karena itu, operator `in` pada dictionary membutuhkan waktu yang sama untuk memprosesnya tanpa memperdulikan jumlah item dalam dictionary.

## 1.2 Dictionary sebagai set penghitung (counters)

Dalam konteks menghitung banyaknya huruf yang muncul dalam sebuah string, penerapan menggunakan model dictionary lebih praktis daripada dua model lainnya yang disebutkan. Ini karena dengan menggunakan dictionary, kita tidak perlu membuat variabel atau list terpisah untuk setiap huruf dalam alfabet. Sebagai gantinya, kita hanya perlu menambahkan huruf-huruf yang muncul sebagai kunci dalam dictionary dan menambah nilai perhitungan yang sesuai. Pendekatan ini lebih efisien dan tidak tergantung pada jumlah karakter dalam alfabet, sehingga lebih fleksibel dan mudah dikelola. Contoh =

```
1 word = 'brontosaurus'
2 d = dict()
3 for i in word:
4     if i not in d:
5         d[i] = 1
6     else:
7         d[i] = d[i] + 1
8 print (d)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER COMMENTS

```
PS D:\PRALPRO\71230997_minggu11> & C:/Users/chris/AppData/Local/Programs/Python/Python38-32/Python.exe -i
{'b': 1, 'r': 2, 'o': 2, 'n': 1, 't': 1, 's': 2, 'a': 1, 'u': 2}
```

Histogram adalah istilah statistika yang mengacu pada set perhitungan atau frekuensi. Dalam konteks komputasi di atas, pendekatan menggunakan perulangan `for` untuk melewati string. Setiap kali terjadi iterasi, jika karakter `i` tidak ada dalam dictionary, item baru dibuat dengan kunci `i` dan nilai awal 1 (mengasumsikan karakter tersebut telah muncul sekali). Jika `i` sudah ada dalam dictionary, nilai `d[i]` diperbarui dengan penambahan.

Output dari program diatas adalah : {'b': 1, 'r': 2, 'o': 2, 'n': 1, 't': 1, 's': 2, 'a': 1, 'u': 2}

Metode `get()` pada dictionary mengambil kunci dan nilai default. Jika kunci muncul dalam dictionary, akan mengembalikan nilai yang sesuai; jika tidak, maka akan mengembalikan nilai default.

Contoh =

```
1 counts = {'chuck' : 1 , 'annie' : 42, 'jan': 100}
2 print(counts.get('jan', 0))
3 print(counts.get('tim', 0))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

```
PS D:\PRALPRO\71230997_minggu11> & C:/Users/chris/AppData/Local/Programs/Python/Python38-32/Python.exe -i
100
0
```

Dengan menggunakan metode `get()`, kita dapat menulis loop histogram secara lebih ringkas. Metode ini secara otomatis menangani kasus di mana kunci tidak ada dalam dictionary. Dengan demikian, kita dapat menghilangkan pernyataan `if` dan merangkum empat baris kode menjadi satu baris saja.

```

1 word = 'brontosaurus'
2 d = dict()
3 for i in word:
4     d[i] = d.get(i,0) + 1
5 print (d)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER COMMENTS

Contoh = `PS D:\PRALPRO\71230997_minggu11> & C:/Users/chris/AppData/Local/Programs/Python/Python311/python.exe d:/PRALPRO/71230997_minggu11/aid.py`  
`{'b': 1, 'r': 2, 'o': 2, 'n': 1, 't': 1, 's': 2, 'a': 1, 'u': 2}`

Output dari program diatas adalah = `{'b': 1, 'r': 2, 'o': 2, 'n': 1, 't': 1, 's': 2, 'a': 1, 'u': 2}`

Penggunaan metode `'get()'` untuk menyederhanakan loop penghitungan telah menjadi idiom yang sangat umum dalam Python. Dibandingkan dengan loop menggunakan pernyataan `'if'` dan operator `'in'`, loop menggunakan metode `'get()'` melakukan hal yang sama, tetapi lebih ringkas.

### 1.3 Dictionary dan File

Dictionary sering digunakan untuk menghitung kemunculan kata-kata dalam sebuah teks, seperti dalam kasus yang disebutkan, menggunakan teks dari Romeo dan Juliet. Kita akan mulai dengan teks yang disederhanakan tanpa tanda baca, lalu beralih ke teks adegan yang menyertakan tanda baca.

```

But soft what light through yonder window breaks
It is the east and Juliet is the sun
Arise fair sun and kill the envious moon
Who is already sick and pale with grief

```

Kita akan menggunakan pola nested loop untuk membaca setiap baris dari file, membagi baris tersebut menjadi daftar kata, dan kemudian menghitung kemunculan setiap kata menggunakan dictionary. Pola ini terdiri dari perulangan luar untuk membaca baris file dan perulangan dalam untuk melakukan iterasi melalui setiap kata dalam baris tersebut.

Dalam pola nested loop, perulangan dalam akan dieksekusi pada setiap iterasi dari perulangan luar. Artinya, perulangan dalam berjalan lebih cepat daripada perulangan luar. Kombinasi dari kedua perulangan ini memastikan bahwa setiap kata pada setiap baris file dihitung.

Dengan menggunakan pola ini, kita dapat menghitung kemunculan kata secara efisien dalam file teks.

```

1 n_f = "romeo.txt"
2 file = open('romeo.txt', 'r')
3
4 counts = dict()
5 for line in file:
6     words = line.split()
7     for word in words:
8         if word not in counts:
9             counts[word] = 1
10        else:
11            counts[word] += 1
12 print (counts)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER COMMENTS

PS D:\PRALPRO\71230997\_minggu11> & C:/Users/chris/AppData/Local/Programs/Python/Python311/python.exe d:/PRALPRO/71230997\_minggu11/aid.py  
`{'But': 1, 'soft': 1, 'what': 1, 'light': 1, 'through': 1, 'yonder': 1, 'window': 1, 'breaks': 1, 'It': 1, 'is': 3, 'the': 3, 'east': 1, 'and': 3, 'Juliet': 1, 'sun': 2, 'Arise': 1, 'fair': 1, 'kill': 1, 'envious': 1, 'moon': 1, 'Who': 1, 'already': 1, 'sick': 1, 'pale': 1, 'with': 1, 'grief': 1}`

Pada bagian `else`, digunakan model penulisan yang lebih singkat untuk menambahkan variabel. `counts[word] += 1` sama dengan `counts[word] = counts[word] + 1`. Metode lain yang dapat digunakan untuk mengubah nilai suatu variabel dengan jumlah yang diinginkan termasuk `-=`, `\*=`, dan `/=`. Ketika program dijalankan, akan diperoleh dump data jumlah kata dalam urutan hash yang tidak disortir.

## 1.4 Looping dan Dictionary

Dalam statement for, dictionary akan bekerja dengan cara menelusuri kunci yang ada didalamnya. Looping ini akan melakukan pecetakan setiap kunci sesuai dengan hubungan nilainya.

Contoh =

```
1 counts = { 'chuck' : 1 , 'annie' : 42, 'jan': 100}
2 for key in counts:
3     print(key, counts[key])
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER COMMENT

PS D:\PRALPRO\71230997\_minggu11> & C:/Users/chris/AppData/Local/Programs/Python/Python39-6/Python.exe -i  
chuck 1  
annie 42  
jan 100

Output menunjukkan bahwa kunci dalam dictionary tidak berada dalam pola urutan tertentu. Pola ini dapat diimplementasikan menggunakan berbagai idiom looping yang telah dijelaskan sebelumnya. Sebagai contoh, jika ingin menemukan semua entri dalam dictionary dengan nilai di atas 10, kode programnya adalah sebagai berikut:

```
1 counts = { 'chuck' : 1 , 'annie' : 42, 'jan': 100}
2 for key in counts:
3     if counts[key] > 10 :
4         print(key, counts[key])
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER COMMENT

PS D:\PRALPRO\71230997\_minggu11> & C:/Users/chris/AppData/Local/Programs/Python/Python39-6/Python.exe -i  
annie 42  
jan 100

Untuk mencetak kunci dalam urutan alfabetis, langkah pertama adalah membuat list dari kunci-kunci dalam dictionary menggunakan metode `keys()` yang tersedia pada objek dictionary. Langkah selanjutnya adalah melakukan pengurutan dengan fungsi `sorted()` pada list tersebut, kemudian melakukan perulangan melalui list yang sudah diurutkan. Selama perulangan, setiap kunci akan dicetak bersama dengan nilai yang sesuai dari dictionary. Implementasi dalam kode dapat dilihat di bawah ini:

```
1 counts = { 'chuck' : 1 , 'annie' : 42, 'jan': 100}
2 lst = list(counts.keys())
3 print(lst)
4 lst.sort()
5 for key in lst:
6     print(key, counts[key])
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER

```
PS D:\PRALPRO\71230997_minggu11> & C:/Users/chris/AppData/Local
['chuck', 'annie', 'jan']
annie 42
chuck 1
jan 100
```

Pertama-tama kita melihat list dari kunci dengan tidak berurutan yang didapatkan dari method kunci. Kemudian kita melihat pasangan nilai kunci yang disusun secara berurutan menggunakan loop for.

### 1.5 Advanced Text Parsing

Pada bagian sebelumnya, kita menggunakan contoh file dimana kalimat pada file tersebut sudah dihilangkan tanda bacanya. Bagian ini kita akan menggunakan file dengan tanda baca lengkap. Dengan menggunakan contoh yang sama yaitu romeo.txt dengan tanda baca lengkap.

```
But, soft! what light through yonder window breaks?
It is the east, and Juliet is the sun.
Arise, fair sun, and kill the envious moon,
Who is already sick and pale with grief,
```

Metode `translate()` dalam Python adalah metode yang digunakan untuk menerapkan terjemahan atau transformasi pada string. Namun, metode ini memiliki beberapa kemungkinan penggunaan yang lebih kompleks dan subtlety (kehalusan) yang tidak selalu terlihat dengan jelas. Saat digunakan untuk memanipulasi string, metode ini biasanya berpasangan dengan metode `maketrans()` yang menciptakan tabel translasi untuk digunakan dengan `translate()`.

Tabel translasi ini menyediakan pemetaan karakter ke karakter lain atau ke None (untuk menghapus karakter). Misalnya, dapat digunakan untuk menghapus tanda baca dari string atau untuk mengubah huruf besar menjadi huruf kecil. Dengan menggunakan `maketrans()` dan `translate()`, string dapat dengan cepat diubah sesuai kebutuhan tanpa perlu perulangan atau metode pemisahan lainnya.

Selain itu, metode ini berguna untuk mempercepat operasi pada string karena operasi translate jauh lebih cepat daripada penggunaan perulangan untuk manipulasi string.

Dengan demikian, metode `translate()` adalah alat yang kuat untuk manipulasi string dalam Python, yang dapat digunakan dengan metode `maketrans()` untuk melakukan berbagai transformasi dengan efisiensi dan kehalusan yang baik.

```
line.translate(str.maketrans(fromstr, tostr, deletestr))
```



Dalam metode `translate()` Python, Anda dapat mengganti karakter dalam string dari satu set karakter ke set karakter lain atau menghapusnya. Dengan menyediakan string kosong sebagai `fromstr` dan `tostr`, dan mengisi `deletestr` dengan tanda baca yang ingin dihapus, Python secara otomatis akan mengidentifikasi dan menghapus tanda baca dari string. Ini merupakan cara efisien untuk membersihkan string dari tanda baca tanpa perlu secara manual menentukan set karakter yang ingin dihapus.

Contoh =

```
import string

n_f = "romeo-full.txt"
file = open('romeo-full.txt', 'r')

counts = dict()
for line in file:
    line = line.rstrip()
    line = line.translate(line.maketrans('', '', string.punctuation))
    line = line.lower()
    words = line.split()
    for word in words:
        if word not in counts:
            counts[word] = 1
        else:
            counts[word] += 1

print (counts)
```

```
PS D:\PRALPRO\71230997_minggu1> & C:/Users/chris/AppData/Local/Programs/Python/python311/python.exe d:/PRALPRO/71230997_minggu1/aid.py
{'romeo': 40, 'and': 42, 'juliet': 32, 'act': 1, '2': 2, 'scene': 2, 'iii': 1, 'capulets': 1, 'orchard': 2, 'enter': 1, 'he': 5, 'jests': 1, 'at': 9, 'scars': 1, 'that': 30, 'never': 2, 'felt': 1, 'a': 24, 'wound': 1, 'appears': 1, 'above': 6, 'window': 2, 'but': 18, 'soft': 1, 'what': 11, 'light': 5, 'through': 2, 'yonder': 2, 'breaks': 1, 'it': 22, 'is': 21, 'the': 34, 'east': 1, 'sun': 2, 'arise': 1, 'fair': 4, 'kill': 2, 'envious': 2, 'moon': 4, 'who': 5, 'already': 1, 'sick': 2, 'pale': 1, 'with': 8, 'grief': 2, 'thou': 32, 'her': 14, 'maid': 2, 'art': 7, 'far': 2, 'more': 9, 'than': 6, 'she': 9, 'be': 14, 'not': 18, 'si nce': 1, 'vestal': 1, 'livery': 1, 'green': 1, 'none': 1, 'fools': 1, 'do': 7, 'wear': 1, 'cast': 1, 'off': 1, 'my': 29, 'lady': 2, 'o': 11, 'love': 24, 'knew': 1, 'were': 9, 'speaks': 3, 'yet': 9, 'says': 1, 'nothing': 1, 'of': 20, 'eye': 2, 'discourses': 1, 'i': 61, 'will': 8, 'answer': 1, 'am': 7, 'too': 8, 'bol d': 1, 'tis': 4, 'to': 34, 'me': 18, 'two': 1, 'fairest': 1, 'stars': 2, 'in': 13, 'all': 6, 'heaven': 3, 'having': 1, 'some': 3, 'business': 1, 'entreat': 1, 'eyes': 6, 'twinkle': 1, 'their': 6, 'spheres': 1, 'till': 4, 'they': 5, 'return': 1, 'if': 12, 'there': 3, 'head': 2, 'brightness': 1, 'cheek': 4, 'would': 16, 'shame': 1, 'those': 2, 'as': 12, 'daylight': 1, 'doth': 2, 'lamp': 1, 'airy': 2, 'region': 1, 'stream': 1, 'so': 10, 'bright': 2, 'birds': 1, 'sing': 1, 'think': 2, 'night': 16, 'see': 2, 'how': 5, 'leans': 1, 'upon': 5, 'hand': 4, 'glove': 1, 'might': 1, 'touch': 1, 'ay': 2, 'speak': 4, 'again': 6, 'angel': 1, 'for': 12, 'glorious': 1, 'this': 9, 'being': 2, 'oer': 1, 'winged': 1, 'messenger': 1, 'unto': 1, 'whiteupturned': 1, 'wondering': 1, 'mortals': 1, 'fall': 1, 'back': 4, 'gaze': 1, 'on': 4, 'him': 2, 'when': 2, 'bestrides': 1, 'lazypacing': 1, 'clouds': 1, 'sails': 1, 'bosom': 1, 'air': 1, 'wherefore': 2, 'deny': 2, 'thy': 20, 'father': 1, 'refuse': 1, 'name': 11, 'or': 4, 'wilt': 6, 'sworn': 1, 'ill': 8, 'no': 6, 'longer': 1, 'capulet': 1, 'aside': 1, 'shall': 6, 'h ear': 2, 'enemy': 2, 'thysself': 1, 'though': 1, 'montague': 5, 'whats': 2, 'nor': 5, 'foot': 2, 'arm': 1, 'face': 2, 'any': 4, 'other': 4, 'part': 2, 'belongi ng': 1, 'man': 2, 'which': 7, 'we': 2, 'call': 3, 'rose': 1, 'by': 14, 'smell': 1, 'sweet': 8, 'call'd': 1, 'retain': 1, 'dear': 7, 'perfection': 1, 'owes': 1, 'without': 1, 'title': 1, 'doff': 1, 'thee': 24, 'take': 3, 'myself': 2, 'word': 4, 'new': 1, 'baptized': 1, 'henceforth': 1, 'thus': 1, 'bescreend': 1, 'stu mblest': 1, 'counsel': 2, 'know': 3, 'tell': 3, 'saint': 2, 'hateful': 1, 'because': 1, 'an': 2, 'had': 1, 'written': 1, 'tear': 2, 'ears': 2, 'have': 13, 'dr unk': 1, 'hundred': 1, 'words': 2, 'tongues': 2, 'utterance': 1, 'sound': 2, 'neither': 1, 'either': 1, 'dislike': 1, 'camest': 1, 'hither': 1, 'walls': 2, 'a re': 3, 'high': 1, 'hard': 1, 'climb': 1, 'place': 2, 'death': 2, 'considering': 1, 'kinsmen': 2, 'find': 2, 'here': 4, 'loves': 3, 'wings': 1, 'did': 3, 'oer perch': 1, 'these': 2, 'stony': 1, 'limits': 1, 'cannot': 1, 'hold': 1, 'out': 2, 'can': 2, 'dares': 1, 'attempt': 1, 'therefore': 3, 'let': 3, 'murder': 1, 'alack': 1, 'lies': 2, 'peril': 1, 'thine': 2, 'twenty': 2, 'swords': 1, 'look': 1, 'proof': 1, 'against': 1, 'enmity': 1, 'world': 3, 'saw': 1, 'nights': 1, 'cloak': 1, 'hide': 1, 'from': 4, 'sight': 1, 'them': 1, 'life': 1, 'better': 1, 'ended': 1, 'hate': 1, 'prorogued': 1, 'wanting': 1, 'whose': 1, 'direction': 1, 'foundst': 1, 'first': 1, 'prompt': 1, 'inquire': 1, 'lent': 2, 'pilot': 1, 'wert': 1, 'vast': 1, 'shore': 1, 'wash'd': 1, 'farthest': 1, 'sea': 2, 'adventu re': 1, 'such': 2, 'merchandise': 1, 'knowst': 1, 'mask': 1, 'else': 3, 'maiden': 1, 'blush': 1, 'bepaint': 1, 'hast': 1, 'heard': 1, 'tonight': 3, 'fain': 3, 'dwell': 2, 'form': 1, 'spoke': 1, 'farewell': 1, 'compliment': 1, 'dost': 2, 'say': 5, 'swearst': 1, 'mayst': 2, 'prove': 4, 'false': 1, 'lovers': 2, 'perju ries': 1, 'then': 2, 'jove': 1, 'laughs': 1, 'gentle': 1, 'pronounce': 1, 'faithfully': 1, 'thinkst': 1, 'quickly': 1, 'won': 1, 'frown': 1, 'perverse': 1, 'n ay': 1, 'woo': 1, 'truth': 1, 'fond': 1, 'havior': 1, 'trust': 1, 'gentleman': 1, 'true': 3, 'cunning': 1, 'strange': 2, 'should': 2, 'been': 1, 'must': 1, 'c onfess': 1, 'overheardst': 1, 'ere': 2, 'was': 1, 'ware': 1, 'passion': 1, 'pardon': 1, 'impute': 1, 'yielding': 1, 'dark': 1, 'hath': 1, 'discovered': 1, 'bl essed': 3, 'swear': 6, 'tips': 1, 'silver': 1, 'fruittree': 1, 'tops': 1, 'inconstant': 1, 'monthly': 1, 'changes': 1, 'circled': 1, 'orb': 1, 'lest': 1, 'lik ewise': 1, 'variable': 1, 'gracious': 1, 'self': 1, 'god': 1, 'idolatry': 1, 'believe': 1, 'hearts': 1, 'well': 2, 'although': 1, 'joy': 2, 'contract': 1, 'ra sh': 1, 'unadvised': 1, 'sudden': 1, 'like': 3, 'lightning': 1, 'cease': 2, 'one': 2, 'lightens': 1, 'good': 9, 'bud': 1, 'summers': 1, 'ripening': 1, 'breath ': 1, 'may': 2, 'beauteous': 1, 'flower': 1, 'next': 1, 'meet': 1, 'repose': 1, 'rest': 2, 'come': 5, 'heart': 1, 'within': 5, 'breast': 2, 'leave': 2, 'unsat isfied': 1, 'satisfaction': 1, 'canst': 1, 'exchange': 1, 'faithful': 1, 'vow': 1, 'mine': 3, 'gave': 1, 'before': 1, 'didst': 1, 'request': 1, 'give': 3, 'wo uldst': 1, 'withdraw': 1, 'purpose': 2, 'frank': 1, 'wish': 1, 'thing': 1, 'bounty': 1, 'boundless': 1, 'deep': 1, 'both': 1, 'infinite': 1, 'nurse': 4, 'call s': 2, 'noise': 1, 'adieu': 1, 'anon': 1, 'stay': 2, 'little': 2, 'exit': 4, 'afear'd': 1, 'dream': 1, 'flatteringsweet': 1, 'substantial': 1, 'reenter': 2, 't
```

## MATERI 2

### Kegiatan Praktikum

#### Kasus 1.1

Buatlah sebuah program yang dapat melakukan generate dan mencetak dictionary yang berisi angka antara 1 sampai n dalam bentuk (x,x\*x)

Contoh:

Input Data = 5

Dcitionary = {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

Pembahasan =

Untuk dapat menyelesaikan kasus pertama, langkah-langkah yang menjadi penyelesaiannya adalah :

1. Membuat dan menentukan panjang dictionary nya.
2. Buat perulangan sepanjang dictionary
3. Input dictionary dengan key = x dan value = x\*x.

```
1 n= int(input("Input data = "))
2 kamus = dict()
3 for x in range(1,n+1):
4     kamus[x]=x*x
5
6 print("Dictionary = ", kamus)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PC

```
PS D:\PRALPRO\71230997_minggu11> & C:/Users/chr
Input data = 5
Dictionary = {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

#### Kasus 1.2

Buatlah program untuk mencetak semua nilai (value) unik yang ada didalam dictionary.

Contoh:

Data : [{"V": "S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"},  
{"VII": "S005"}, {"V": "S009"}, {"VIII": "S007"}]

Output : Unique Values: {'S005', 'S002', 'S007', 'S001', 'S009'}

## Pembahasan Kasus 2

Untuk dapat menyelesaikan kasus kedua, kita dapat menggunakan fungsi values pada dictionary. untuk mencari nilai unik kita ambil masing-masing nilai dari anggota dictionary, kemudiah kita kumpulkan dalam satu variabel.

```
1 data = [{"V": "S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII": "S005"}, {"V": "S009"}, {"VIII": "S007"}]
2 print("Data asli: ", data)
3 nilai_unik = set( val for dic in data for val in dic.values())
4 print("Nilai Unik: ", nilai_unik)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER COMMENTS

PS D:\PRALPRO\71230997\_minggu11> & C:/Users/chris/AppData/Local/Programs/Python/Python311/python.exe d:/PRALPRO/71230997\_minggu11/a  
Data asli: [{"V": 'S001'}, {'V': 'S002'}, {'VI': 'S001'}, {'VI': 'S005'}, {'VII': 'S005'}, {'V': 'S009'}, {'VIII': 'S007'}]  
Nilai Unik: {'S007', 'S009', 'S001', 'S005', 'S002'}

### Kasus 1.3

Dengan menggunakan file words.txt, buatlah program untuk menyimpan kunci (keys) pada dictionary. Tambahkan pengecekan apakah suatu kata yang diinputkan ada didalam daftar tersebut. Jika ada silakan cetak kata ditemukan, jika tidak ada silakan cetak kata tidak ditemukan.

Masukkan nama file : words.txt

Kata yang dicari : programming

Daftar Kamus

```
{'Writing': 1, 'programs': 2, 'or': 3, 'programming':  
4, 'is': 5, 'a': 6, 'very': 7, 'creative': 8, 'and':  
9, 'rewarding': 10, 'activity': 11, 'You': 12,  
'can': 13, 'write': 14, 'for': 16, 'many': 17,  
'reasons': 18, 'ranging': 19, 'from': 20, 'making':  
21, 'your': 22, 'living': 23, 'to': 24, 'solving':  
25, 'difficult': 27, 'data': 28, 'analysis': 29, }
```

## Pembahasan Kasus 3

Untuk dapat membuat program diatas, langkah-langkah yang harus dilakukan adalah :

1. Buat program untuk membaca file txt.
2. Input kata yang dicari.
3. Buat dictionary baru untuk menyimpan file yang sudah dibaca.
4. simpan file dalam dictionary, jangan lupa cek duplikasinya.
5. Cetak dictionary nya.
6. Lakukan pengecekan terhadap kata yang diinputkan.

7. Cetak hasilnya.

```
count = 0
dictionary_words = dict()
fname = input('Masukkan nama file : ')
fword = input('Kata yang dicari : ')
try:
    fhand = open(fname)
except FileNotFoundError:
    print('File tidak bisa dibuka !!', fname)
    exit()

for line in fhand:
    words = line.split()
    for word in words:
        count += 1
        if word in dictionary_words:
            continue
        dictionary_words[word] = count
print('\nDaftar Kamus : \n')
print(dictionary_words)
if fword in dictionary_words:
    print('\nKata %s ditemukan dalam kamus' % fword)
else:
    print('\nKata %s tidak ditemukan dalam kamus' % fword)
```

## BAGIAN 2: LATIHAN MANDIRI (60%)

Pada bagian ini anda menuliskan jawaban dari soal-soal Latihan Mandiri yang ada di modul praktikum. Jawaban anda harus disertai dengan source code, penjelasan dan screenshot output.

### SOAL 1

```
dictionary = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

print("key", "value", "item", sep='\t')

for key, value in dictionary.items():
    print(key, value, key, sep='\t')
```

✓ 0.0s

| key | value | item |
|-----|-------|------|
| 1   | 10    | 1    |
| 2   | 20    | 2    |
| 3   | 30    | 3    |
| 4   | 40    | 4    |
| 5   | 50    | 5    |
| 6   | 60    | 6    |

Membuat kamus dengan variabel bernama dictionary dengan elemen elemen berisi beberapa angka yang sama, lalu mengprint judul seperti key, value, item lalu dirapikan dengan sep=\t. lalu menggunakan loop for untuk mengakses setiap pasangan nilai dalam kamus. Metode .items() digunakan untuk mendapatkan pasangan kunci-nilai dalam bentuk tuple. Lalu mengprint isi key, value, dan key lagi dari setiap item dan dirapikan dengan sep=\t.

## SOAL 2

```
list1 = []
while True:
    isi = input('masukan isi list / ketik "selesai" untuk lanjut ke list 2 = ')
    if isi.lower() == 'selesai':
        break
    else:
        list1.append(isi)
list2 = []
while True:
    isi2 = input('masukan isi list / ketik "selesai" untuk seelsai = ')
    if isi2.lower() == 'selesai':
        break
    else:
        list2.append(isi2)

result_dict = dict(zip(list1, list2))
print (result_dict)
```

✓ 46.1s

```
{'red': '#FF0000', 'green': '#008000', 'blue': '#0000FF'}
```

Membuat list untuk pengguna aka nisi, lalu menggunakan perulangan tak terbatas akan meminta input pengguna hingga pengguna memasukkan input 'selesai' maka akan lanjut untuk mengisi list2 hingga pengguna menginput 'selesai'. Lalu akan membuat variabel baru untuk menggabungkan 2 list menjadi satu dan menggunakan fungsi dict() untuk membuat kamus. Lalu akan mengprint variable baru tersebut berisi 2 list yang dijadikan satu.

## SOAL 3

```
n_f = "mbox-short.txt"
file = open('mbox-short.txt', 'r')

h_e = {}
for i in file:
    if i.startswith('From '):
        kata = i.split()
        email = kata[1]
        h_e[email] = h_e.get(email, 0) + 1
file.close()

print(h_e)
```

✓ 0.1s

Python

```
{'stephen.marquard@uct.ac.za': 2, 'louis@media.berkeley.edu': 3, 'zqian@umich.edu': 4, 'rjlowe@iupui.edu': 2, 'cwen@iupui.edu': 5, 'gsilver@umich.edu': 3, 'wagner
```

Membuka file mbox-short.txt lalu akan membaca nya, lalu membuat kamus kosong bervariasi h\_e untuk menyimpan Alamat email dan jumlah kemunculannya. Lalu menggunakan loop for untuk mengloop tiap isi dari file mbox-short.txt menggunakan .startswith() untuk mencari kata kata yang berumulai dengan 'From' lalu dalam variable kata akan mengsplit tiap kata dan menyimpannya dalam variabel kata lalu mengambil alamat email dari indeks kedua di dalam list kata. Dalam format 'From ', alamat email berada pada indeks kedua setelah kata 'From'. Menggunakan metode .get() untuk mendapatkan nilai dari kamus h\_e untuk kunci email. Jika kunci email belum ada dalam kamus, maka get() akan mengembalikan nilai default 0. Kemudian, jumlahnya ditambah 1 dan disimpan kembali

dalam kamus `h_e` untuk kunci email. Lalu menutup file setelah selesai membacanya dan mengprint '`h_e`'.

#### SOAL 4

```
n_f = "mbox-short.txt"
file = open('mbox-short.txt', 'r')

hitung = {}
for i in file:
    if i.startswith('From '):
        kata = i.split()
        email = kata[1]
        hm = email.find('@')
        ha = email[hm + 1:]
        hitung[ha] = hitung.get(ha, 0)+1
file.close()

print(hitung)
```

✓ 0.0s

```
{'uct.ac.za': 6, 'media.berkeley.edu': 4, 'umich.edu': 7, 'iupui.edu': 8, 'caret.cam.ac.uk': 1, 'gmail.com': 1}
```

Membuka file `mbox-short.txt` lalu akan membacanya, Membuat kamus kosong '`hitung`' yang akan digunakan untuk menyimpan domain email dan jumlah kemunculannya. Menggunakan loop `for` untuk membaca tiap kata di `mbox-short.txt` lalu memeriksa apakah kata kata dimulai dari kata '`From`' lalu Memisahkan baris menjadi potongan kata-kata dan menyimpannya dalam list '`kata`'. Mengambil alamat email dari indeks kedua di dalam list '`kata`' lalu mencari posisi karakter '@' dalam alamat email dan menyimpannya dalam variabel `hm`. Memisahkan domain dari alamat email dengan cara mengambil substring mulai dari posisi '`hm + 1`' hingga akhir alamat email dan menyimpannya dalam variabel `ha`. Menggunakan metode `.get()` untuk mendapatkan nilai dari kamus `hitung` untuk kunci `ha`. Jika kunci `ha` belum ada dalam kamus, maka `get()` akan mengembalikan nilai default 0. Kemudian, jumlahnya ditambah 1 dan disimpan kembali dalam kamus `hitung` untuk kunci `ha`. Lalu menutup file setelah selesai membaca lalu akan mengprint variabel '`hitung`'.

LINK GITHUB = <https://github.com/Yeeemeki/praktikum-algoritma-dan-pemograman.git>