

LAPORAN AKHIR SEMESTER

Pemrograman Berbasis
Fungsi

DIBERIKAN KEPADA

Riksa Meidy Karim, S.Kom., M.Si.,
M.Sc.

DISUSUN OLEH

Alfa Khoirin

Jurnal Praktikum

Modul 1

Seorang mahasiswa sains data ingin menyewa buku dari sebuah startup yang menyediakan layanan sewa buku. Startup tersebut memiliki ketentuan sewa dengan aturan sebagai berikut:

- Harga sewa buku berbeda-beda sesuai dengan kategorinya
- Harga sewa buku dihitung berdasarkan jumlah halaman nya
- Harga sewa buku dihitung per hari nya
- Maksimal durasi sewa adalah 26 hari

Startup tersebut masih dalam tahap awal pengembangan, sehingga ingin melakukan uji coba penyewaan 5 kategori buku. Berikut rincian kategori nya:

- Kategori 1 : 100 rupiah per lembar per hari
- Kategori 2 : 200 rupiah per lembar per hari
- Kategori 3 : 250 rupiah per lembar per hari
- Kategori 4 : 300 rupiah per lembar per hari
- Kategori 5 : 500 rupiah per lembar per hari

Startup tersebut memerlukan sebuah program untuk:

- menghitung total biaya dari customer
- mencatat tanggal awal sewa, dan durasi hari
- menampilkan informasi kapan tanggal pengembalian buku dari customer

Format input tanggal adalah yyyy-mm-dd. Bantulah startup tersebut membuat program tersebut dengan menggunakan konsep modularisasi!

```
In [1]: kategoris = {
        1: 100,
        2: 200,
        3: 250,
        4: 300,
        5: 500,
        }

tanggal = input("Tanggal sewa: ") #format yyyy-mm-dd
durasi = int(input("Masukkan lama waktu sewa:"))
```

Tanggal sewa: 2022-06-16
Masukkan lama waktu sewa:10

```
In [2]: def dtl (s_tgl):
        return [ int(k) for k in s_tgl.split('-')]

def is_cm (tgl_p,d,c):
    return tgl_p[2]+ d > c

def thn_back (tgl_p,d,c):
    return tgl_p[0]+1 if ( is_cm(tgl_p,d,c) and tgl_p[1] == 12) else tgl_p[0]

def bln_back (tgl_p,d,c):
    return ( tgl_p[1] % 12 )+1 if is_cm(tgl_p,d,c) else (tgl_p[1])

def tgl_back (tgl_p,d,c):
    return tgl_p[2] + d - c if is_cm(tgl_p,d,c) else tgl_p[2] + d

def is_awal_abad(thn):
    return thn % 100 == 0

def kabisat (thn):
    return(is_awal_abad(thn) and thn % 400 == 0) or (not is_awal_abad(thn) and thn % 4 == 0)

def dec_c(t):
    return 30 +( t[1]%2 if t[1]<= 8 else abs((t[1]%2)-1)) if t[1] != 2 else(29 if kabisat(t[0]) else 28)

def wkt_kembali (tgl_p,d):
    return [thn_back(tgl_p,d, dec_c(tgl_p)),bln_back(tgl_p,d, dec_c(tgl_p)),tgl_back(tgl_p,d, dec_c(tgl_p))]
```

```
In [3]: tgl_p = dtl(tanggal)
        wkt_kembali(tgl_p,durasi)

Out[3]: [2022, 6, 26]

In [4]: hari = dec_c(tgl_p)
        print(hari)

30

In [5]: sewaan_all = [ [1,5], [2,3], [3,0], [4,1], [5,2] ]

        def calc_biaya_per_kategori(kategoris, sewaan):
            return sewaan[1] * kategoris.get(sewaan[0])

        def calc_all_biaya(kategoris, sewaan_all, durasi):
            return sum([calc_biaya_per_kategori(kategoris, sewaan) for sewaan in sewaan_all]) * durasi

In [6]: calc_all_biaya(kategoris, sewaan_all, durasi)

Out[6]: 24000
```

Modul 2

Kerjakan seluruh soal berikut dengan menggunakan higher order function map,filter dan reduce:

1. Buatlah sebuah fungsi bernama ulang_i_NIM, ulang_i memiliki input sebuah bilangan skalar a, dan mengeluarkan vektor 1xn dengan seluruh elemen nya adalah a !

```
In [1]: a = 15

        def ulang_i_015(a):
            return list(map(lambda x: a, range(a)))

        ulang_i_015(a)

Out[1]: [15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15]
```

2. Buatlah deret bilangan sebagai berikut dengan input n sebagai panjang deret:

$$\frac{1}{2}, -\frac{1}{4}, \frac{1}{8}, \dots, (-1)^n \frac{1}{2^{n+1}}$$

```
In [51]: n = int(input("Masukkan jumlah bilangan deret : "))
        a = list(range(1,n+1))

        def deret(n):
            return ((-1)**(n+1)) * (1/(2**n))
            print(list(map(deret,a)))

        Masukkan jumlah bilangan deret : 15
        [0.5, -0.25, 0.125, -0.0625, 0.03125, -0.015625, 0.0078125, -0.00390625, 0.001953125, -0.0009765625, 0.00048828125, -0.000244140625, 0.0001220703125, -6.103515625e-05, 3.0517578125e-05]
```

3. Jumlahkan deret bilangan tersebut!

```
In [52]: from functools import reduce as r
        def add(a,b):
            res = a+b
            print('a:',a, ' b:', b, '-> a+b:',res)
            return res
        print(r( add, list(map(deret,a)) ))

        a: 0.5 , b: -0.25 -> a+b: 0.25
        a: 0.25 , b: 0.125 -> a+b: 0.375
        a: 0.375 , b: -0.0625 -> a+b: 0.3125
        a: 0.3125 , b: 0.03125 -> a+b: 0.34375
        a: 0.34375 , b: -0.015625 -> a+b: 0.328125
        a: 0.328125 , b: 0.0078125 -> a+b: 0.3359375
        a: 0.3359375 , b: -0.00390625 -> a+b: 0.33203125
        a: 0.33203125 , b: 0.001953125 -> a+b: 0.333984375
        a: 0.333984375 , b: -0.0009765625 -> a+b: 0.3330078125
        a: 0.3330078125 , b: 0.00048828125 -> a+b: 0.33349609375
        a: 0.33349609375 , b: -0.000244140625 -> a+b: 0.333251953125
        a: 0.333251953125 , b: 0.0001220703125 -> a+b: 0.3333740234375
        a: 0.3333740234375 , b: -6.103515625e-05 -> a+b: 0.33331298828125
        a: 0.33331298828125 , b: 3.0517578125e-05 -> a+b: 0.333343505859375
        0.333343505859375
```

4. Sebuah DNA dimodelkan dalam sebuah string menjadi sequence TCGA dan disimpan ke dalam data :

<https://drive.google.com/file/d/18C1ylSTXrY9pglqqhijoS8LYmcxdIjM/view?usp=sharing>

hitunglah jumlah kemunculan pola berikut pada data tersebut:

```
In [12]: seq = 'A'
def append_n(file,i,n):
    return file[i:i+n] #men genrate huruf dari file berdasarkan len() dari huruf yang ingin dicari

def remap(file,seq):
    return map(lambda x: append_n(file,x,len(seq)), range(len(file) - len(seq)+1)) # me-remap/ membuat ulang list berdasarkan fu

def count_mer(file,seq):
    return r(lambda a,b: a +(1 if b == seq else 0), remap(file,seq),0) # menghitung jumlah isi List yang sama dengan sequence.

In [13]: a = 0
append_n(file,a,len(seq)) # Menampilkan cut yang akan ditambahkan sebagai contoh saya buat i = 0 atau list yang menjadi urutan pe

Out[13]: 'T'
```

A. A

```
In [2]: import re
import string
obj_file = open("DNA.txt", "r")

file = obj_file.read()
file = file[:-1]
file

Out[2]: 'TGCTCTCCGGCTGAGCGGTTCTTAACCGAGAGCTGATACTGGTCGAATATCGACGGGCAAGAGCCCTGGGATTGATGCGTTTACCATGCGCGTCTCAGTGCAGGCAGGAATGCAGAGCTTACTT
CAAACTAGTTACTGGCAAAAATACAAATTTTTCGATCGACCTTGAGTTTATTATTACCGCACAGTCTTTTACCGCACCTGTTACCGCACATCCGTAAGTTTACCGCACGTTACCGCACTACCTC
TCTATATTACCGCACTTCGTTTACCGCACGCTGAGGAACGGTTACCGCACTTACCGCACCAAGGTGCGTGCTCTGTTATTACCGCACCACTTACCGCACGACCTTTTATTACCGCACCAAGGGC
ACAGCCACGTAGGGTAGCGTCGTTCTCACTGTATTGCGGCACGGTCGTAATTTACCGCATTACCGCACCACTCGTTAGCTTACCGCACCTAGGGTTGTTACCGCACGACTTACCGCACAGCCGTTA
```

B. AT

```
In [18]: seq_AT = 'AT'
print('Jumlah pola AT yang terjadi sebanyak : ',count_mer(file,seq_AT))

Jumlah pola AT yang terjadi sebanyak : 557
```

C. GGT

```
In [19]: seq_GGT = 'GGT'
print('Jumlah pola GGT yang terjadi sebanyak : ',count_mer(file,seq_GGT))

Jumlah pola GGT yang terjadi sebanyak : 77
```

D. AAGC

```
In [23]: seq_AAGC = 'AAGC'
print('Jumlah pola AAGC yang terjadi sebanyak : ',count_mer(file,seq_AAGC))

Jumlah pola AAGC yang terjadi sebanyak : 22
```

E. AGCTA

```
In [22]: seq_AGCTA = 'AGCTA'
print('Jumlah pola AGCTA yang terjadi sebanyak : ',count_mer(file,seq_AGCTA))

Jumlah pola AGCTA yang terjadi sebanyak : 5
```

5. Reverse complement dari suatu sequence string DNA memiliki aturan sebagai berikut:

A adalah komplemen dari T

C adalah komplemen dari G

Contoh reverse complement:

input DNA : ACTGA

Reverse complmenet : TGACT

Buatlah fungsi untuk mencari inverse komplemen dari data pada nomor 4 !

```
In [24]: def komplemen(x):
    return {'A':'T', 'T':'A', 'C':'G', 'G':'C'}.get(x) # Mencari dan mendapatkan value yang sesuai pada directory

def reverse_kom(file):
    return map(lambda x:komplemen(x), file)# membuat ulang List (map()) yang telah di tukar valuenya sesuai dengan yang ada pada

In [25]: res = reverse_kom(file)
print(*res)

A C A G A A G G C C G A C T C G C C A A G G A T T G G T C G T C T G A C T A T G A C C A G C T T A T A G C T G C C C G T T C T
C G G G A C C C T A A C T A C G C A A A G T G G T A C G C G C A G A G T C A C G T C C G T C C T T A C G T C T C G A A T G A A
G T T T G A T C A A T G A C C G T T T T T A T G T T A A A A A A G C T A G C T G G A A C T C A A A T A A G T A A T G G C G
T G T C A G A A A A T G G C G T G G A C A A T G G C G T G T A G G C A T T C A A A T G G C G T G C A A T G G C G T G A T G G A
G A G A T A T A A T G G C G T G A A G C A A A T G G C G T G C G A C T C C T T G C C A A T G G C G T G A A T G G C G T G G T G
T T C C A C G C A C G A G A C A A T A A T G G C G T G G T G G T A A T G G C G T G C G T G A A A A T A A T G G C G T G G T C C
C G T G T C G G T G C A T C C C A T C G C A G C A A G A G T G A C A T A A C G C C G C T G C C A G C A T T A A A T G G C G T A
A T G G C G T G G T G A G C A A T C G A A T G G C G T G G A T C C C A A C A A T G G C G T G C T G A A T G G C G T G T C G G C
A A T G G C G T G C A C A A T G A A C T G C G A G A T T G A G G G T G A G T A T A G T C A G A A T A A T G G C G T G T G A C C
C G A A T G G C G T G G G C G T G G A A T T C A T C C G T C A A T G G C G T G C A T A A T G G C G T G C A T A A T G G C G T
G T G G A C A T T T C C G T C C C A T T T C A T G T C T G A A T G G C G A A T G G C G T G C C A A C G T G G T G C T G T T A
G A T T G C A A T C C A T G C A A T G G C G T G C C C T T T A A T G G C G T G A G G T C C C A A A A T G G C G T G T C T A T A
G G T A A G C C C T T A C A C T G G G G A C C T C A C C T C A A C A C G C T T T C T A T G C C T C A A A A G T T C C G T G T
G G G T C G A T A C A A T A A T T C G C A A T G T C A C C G G C G A C G T A G T A C A G T T A C A A G T C C A G T A A G A G A
T A G A A C G A T A C A T G C T T G G G A G C A A T T C T C C T C A T T C G C T A G A A A A C T G T T T T A G C A T A C G T
A C A T C C G C T C C G T T A C G G C T A A T G T A A C T T G C C G C C C T G A A A A G C A T A C T C T G T G G C G C C A A C
T T T A T A A A A A A A T A C G T T C T C G C C C T A A C C C G C C T T C C T C T G A A T T G C G T C A C G G A T C G T G A C
A A T T G A C G C G T A C C G G C C T A C T G A T G G A T A A A A C G T C G A G G T C G C A A A C T C A A G G T G C A T G
A C T G C C T T G T C A G G G C T C T A T C C G G T A C A C C A G C T A G G G T C A C T C T T T A C T C T G A G C T C T A C G
```

6. Terdapat proses yang dinamakan feed-forward. Input dalam sebuah neural network diproses ke hidden layer hingga ke output layer.

Setiap Node, menunjukkan neuron dan setiap garis menunjukkan weight.

Proses Feed-Forward berjalan dari input layer menuju output layer.

Nilai yang masuk ke neuron di hidden layer adalah penjumlahan antara perkalian weight dengan nilai yang masuk pada input neuron setelah itu diaktifkan dengan fungsi aktivasi.

Atau dapat dimodelkan sebagai berikut:

$$S_1 = X_1.W_{11} + X_2.W_{21} + X_3.W_{31}$$
$$S_2 = X_1.W_{12} + X_2.W_{22} + X_3.W_{32}$$
$$A_1 = \frac{1}{1 + e^{-S_1}}$$
$$A_2 = \frac{1}{1 + e^{-S_2}}$$
$$Z_1 = M_{11}.A_1 + M_{21}.A_2$$
$$Y_1 = \frac{1}{1 + e^{-Z_1}}$$

Buatlah fungsi feed-forward dengan input berikut:

W11 = 0.5

W12 = 0.4

W21 = 0.3

W22 = 0.7

W31 = 0.25

W32 = 0.9

M11 = 0.34

M21 = 0.45

dan X1 = 9 , X2 = 10 , X3 = -4

```
In [61]: import math

def aktivasi(x): # Rumus mendapatkan nilai aktivasi A dan Z
    return 1/(1+ math.exp(-x))

def WTi(W,i):
    return list(map(lambda w:w[i], W))

def WT(W):
    return list(map(lambda i: WTi(W,i), range(len(W[0])))) #Membuat list baru yg berisi W ke 1 dan juga W ke 2 yang nantinya akan

def XW(X,W):
    return map(lambda w: r(lambda a,b:a+b , map(lambda xx,ww: xx * ww, X,w),0), WT(W)) # Digunakan Untuk mencari S1, S2, dan Z1

def input_to_hidden(X,W):
    return list(map(lambda x:aktivasi(x), XW(X,W))) #Mendapatkan nilai A1 dan A2 setelah didapatnya nilai S1 dan S2

def feed_forward(X,W,M):
    return input_to_hidden(input_to_hidden(X,W),M) #Mendapatkan hasil Y dengan Z dicari dengan Fungsi XW
```

```
In [62]: X = [9, 10, -4]
         W = [[0.5,0.4],[0.3,0.7],[0.25,0.9]]
         M = [[0.34],[0.45]]

         feed_forward(X,W,M)
```

```
Out[62]: [0.6876336740661236]
```

```
In [69]: A12 = input_to_hidden(X,W) # nilai aktivasi A1 dan A2
         A12
```

```
Out[69]: [0.998498817743263, 0.9990889488055994]
```

```
In [64]: list(Xw(X,W)) #Menghitung Nilai S1 dan S2
```

```
Out[64]: [6.5, 7.0]
```

```
In [65]: WT(W) # hasil transformasi w1,2,3(1) dan w1,2,3(2)
```

```
Out[65]: [[0.5, 0.3, 0.25], [0.4, 0.7, 0.9]]
```

```
In [71]: WTi(W,0)
```

```
Out[71]: [0.5, 0.3, 0.25]
```

Latihan Soal

Pertemuan 9

Buat program untuk menghitung deret bilangan prima dari 2 hingga N menggunakan HOF filter dan map. Contoh primes(100):

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

```
▷ ~  
faktor = lambda n : list(filter(lambda i : n%i==0, range(1,n+1)))  
primes = lambda n : filter (lambda i : len(faktor(i))== 2, range (1,n+1))  
  
print (list(primes(100)))  
[45] ✓ 0.4s  
... [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

```
▷ ~  
l = lambda n : range(2,n)  
faktor = lambda n : list(filter(lambda i:n%i==0, l(n)))  
primes = lambda n : filter (lambda i:len(faktor(i))==0, range(2,n+1))  
  
print (list(primes(100)))  
[46] ✓ 0.3s  
... [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

```
employee = {  
'Nagao':35,  
'Ishii':30,  
'Kazutomo':20,  
'Saito':25,  
'Hidemi':29  
}
```

Terdapat dictionary employee berisi nama dan umur pegawai, lakukan filter untuk mengetahui pegawai yang berumur > 25 tahun !

```
employee = {  
'Nagao':35,  
'Ishii':30,  
'Kazutomo':20,  
'Saito':25,  
'Hidemi':29  
}  
[47] ✓ 0.4s
```

```
▷ ~  
print(list(filter(lambda x:x[1]>25, employee.items())))  
[48] ✓ 0.5s  
... [('Nagao', 35), ('Ishii', 30), ('Hidemi', 29)]
```

```
filter_bye_age = lambda age, employee: list(filter (lambda x:x[1]>25, employee.items() ))  
nama_hasil = map(lambda d:d [0], filter_bye_age(25,employee))  
print(list(nama_hasil))  
[49] ✓ 0.3s  
... ['Nagao', 'Ishii', 'Hidemi']
```


Pertemuan 10

Buat fungsi mencari jumlah bilangan genap dari list L!

Contoh:

L = [2,1,9,10,3,90,15]

```
[8] from functools import reduce as r
✓ 0.8s

> ▾
L = [2,1,9,10,3,90,15]
r(lambda a,b:a + (1 if b % 2 == 0 else 0), L,0)
[33] ✓ 0.1s

... 3
```

Buat fungsi untuk menghitung n! Menggunakan reduce!

```
▷ ▾
faktorial = lambda n : r (lambda a,b : a*b if b>1 else 1, range(1, n+1),1)
for i in range (6+1):
    print(str(i)+ '!', faktorial(i))
[50] ✓ 0.5s

... 0! 1
1! 1
2! 2
3! 6
4! 24
5! 120
6! 720
```

Hitung euclidian distance dari dua vektor berikut menggunakan higher order function!

X = [2,5,6,7,10]

Y = [-2,9,2,-1,10]

```
▷ ▾
X = [2,5,6,7,10]
Y = [-2,9,2,-1,10]

euclid = lambda X,Y : r (lambda a,c:a+c, map( lambda x,y : (x-y)**2 ,X,Y))** 0.5
euclid(X,Y)
[13] ✓ 0.8s

... 10.583005244258363
```


Terdapat dictionary employee berisi nama dan umur pegawai, lakukan reduce untuk mengetahui

berapa jumlah pegawai yang berumur > 25 tahun !

```
employee = {  
'Nagao':35,  
'Ishii':30,  
'Kazutomo':20,  
'Saito':25,  
'Hidemi':29  
}
```

```
employee = {  
    'Nagao':35,  
    'Ishii':30,  
    'Kazutomo':20,  
    'Saito':25,  
    'Hidemi':29  
}  
  
jml_pgwai = lambda umur, employee : r ( lambda a,b : a+1 if b [1]> umur else a, employee.items(),0)  
jml_pgwai(25,employee)  
[51] ✓ 0.4s  
... 3
```

Buatlah deret fibonacci menggunakan higher order function!

```
fibonacci = lambda n : r (lambda a,b : a if b [0]<=1 else a +[ a[b[0]-1]+a[b[0]-1]+ a[b[0]-2]],  
    enumerate ( [0,1] + list( range (1,n))),[0,1]) if n >0 else [0]  
  
for i in range (6):  
    print('fibonacci',i, '->', fibonacci(i))  
[52] ✓ 0.3s  
... fibonacci 0 -> [0]  
    fibonacci 1 -> [0, 1]  
    fibonacci 2 -> [0, 1, 2]  
    fibonacci 3 -> [0, 1, 2, 5]  
    fibonacci 4 -> [0, 1, 2, 5, 12]  
    fibonacci 5 -> [0, 1, 2, 5, 12, 29]
```

Pertemuan 11

Buat sebuah program untuk membuat deret fibonacci dari 0 hingga N dengan menggunakan fungsi non-rekursif dan rekursif!

Bandingkan keduanya jika nilai N = 500, Manakah yang lebih baik? Jelaskan!

Jawab :

Penjelasan, hasil yang saya dapat menggunakan rekursif dan non-rekursif untuk kasus deret fibonacci yaitu perbedaan waktu yang didapat antara keduanya. Untuk deret dengan jumlahnya sedikit mungkin tidak terlalu jauh perbedaannya. Sedangkan, untuk yang jumlah deretnya sudah sangat banyak seperti contoh 500 maka rekursif akan memakan waktu yang sangat – sangat lama karena rekursif melakukan perulangan dan perhitungannya dari awal kembali sedangkan non-rekursif tidak melakukan perulangan dari awal tetapi melanjutkan perhitungan dari data yang telah yang baru sehingga lebih efisien.

Pertemuan 12

Ubah fungsiku menjadi pure function!

```
def fungsiku (L):  
    cL = L.copy()  
    def check_genap(l):  
        return l % 2 == 0  
  
    for i in range (len(L)):  
        if check_genap(L[i]):  
            L[i] = L[i]/2  
        else :  
            L[i] = L[i]*n + 1  
    return L
```

3] ✓ 0.6s

```
n = 3  
L = [5,6,7,8]  
print(fungsiku(L))
```

4] ✓ 0.5s

• [16, 3.0, 22, 4.0]

```
print(L)
```

5] ✓ 0.5s

• [16, 3.0, 22, 4.0]

Ubah fungsiku2 menjadi pure function!

```
def fungsiku2 (L,n):
    cL = L.copy()
    def check_faktor(l):
        return l % n == 0
    for i in range (len(L)):
        if check_faktor(L[i]):
            L[i] = L[i]/2
        else :
            L[i] = L[i]*n + 1
    return L
```

✓ 0.8s

```
n = 3
L = [5,6,7,8]
print(list(fungsiku2(L,n)))
print(L)
```

[16, 3.0, 22, 25]

[16, 3.0, 22, 25]

Apakah isi dalam tupel tup ada yang dapat diubah?

tup = ([3, 4, 5], 'myname')

```
tup = ([3, 4, 5], 'myname')
tup[3]="alfa"
```

```
print("tup 2 :", tup)
```

0.6s

```
-----
TypeError                                Traceback (most recent call last)
c:\Users\Lenovo\Documents\Sains Data Semester 4\Pemogramman berbasis fungsi\Laporan Akhir\latihan_tiapkelas.ipynb Cell 35' in <cell line: 2>()
      1 tup = ([3, 4, 5], 'myname')
----> 2 tup[3]="alfa"
      3 print("tup 2 :", tup)

TypeError: 'tuple' object does not support item assignment
```

Pertemuan 13

Addku = lambda x: x + 10

Powku = lambda x: x**2

Kurku = lambda x: x - 2 * x

a. Buatlah fungsi komposisi menggunakan 3 fungsi diatas yang melakukan hal sebagai berikut secara berurut:

1. Menjumlahkan input dengan nilai 10
2. Menjumlahkan input dengan nilai 10
3. Mengeluarkan nilai kuadrat dari input nya

b. Buatlah fungsi invers nya!

```
addku = lambda x: x+10
powku = lambda x: x**2
kurku = lambda x: x-2*x

f_komp = lambda f,g: lambda x: f(g(x))

my_f_kom = f_komp(kurku,f_komp(powku,addku))

● my_f_kom(15)
```

[6] ✓ 0.7s

... -625



```
#invers
inv_addku = lambda x: x-10
inv_powku = lambda x: x**0.5
inv_kurku = lambda x: -1 * x

my_f_kom_inv = f_komp(inv_addku,f_komp(inv_powku,inv_kurku))
my_f_kom_inv(my_f_kom(15))
```

[7] ✓ 0.5s

... 15.0

Universitas di Lampung ITARE, ingin memiliki sistem penentuan golongan UKT dan jumlah biaya UKT yang dibayarkan oleh Mahasiswa berdasarkan Kriteria berikut:

1. Jumlah tanggungan
2. Jumlah token listrik selama 3 bulan terakhir
3. Gaji Orang tua / Penanggung jawab
4. Penerima program KIP-K atau bukan

Ketentuan:

1.Ketentuan Jumlah Tanggungan :

Jika lebih ≥ 5 , maka skor = 1

Jika < 5 , maka skor = $5 - \text{jumlah tanggungan}$

2.Ketentuan token listrik:

Jika rata-rata lebih dari 100 ribu per bulan , maka skor = 3

Jika diantara 50 ribu - 100 ribu per bulan , maka skor = 2

Jika dibawah 50 ribu , maka skor = 1

3.Ketentuan Gaji :

Jika gaji penanggung jawab > 10 juta maka skor = 7

Jika $8 < \text{gaji} \leq 10$ juta, maka skor = 6

Jika $6 < \text{gaji} \leq 8$ juta , maka skor = 5

Jika $4 < \text{gaji} \leq 6$ juta , maka skor = 4

Jika $3 < \text{gaji} \leq 4$ juta , maka skor = 3

Jika gaji < 3 juta , maka skor = 2

4.Jika mahasiswa memiliki KIP-K , maka skor = 1, jika tidak maka skor = 5

Perhitungan pembayaran UKT adalah sebagai berikut:

$\text{Skor_total} = 20 \% * \text{skor_1} + 30 \% * \text{skor_2} + 20\% \text{ skor_3} + 30\% \text{ skor_4}$

$\text{Jumlah bayar UKT} = \text{biaya pokok} + \text{skor_total} * 500 \text{ ribu}$

Uang Pokok = 750 ribu

Gunakan fungsi komposisi untuk menyelesaikan masalah tersebut!


Hitung berapa biaya UKT yang harus dibayarkan dengan input sebagai berikut:

1.Jumlah tanggungan = 3

2.Listrik 3 bulan terakhir = 120 ribu , 75 ribu , 50 ribu

3.Gaji Penanggung jawab = 5.5 juta per bulan

4.Peserta KIP-K = Tidak

```
> 
def jml_tanggungan(tanggungan):
    skor_1 = 0
    if tanggungan >=5:
        skor_1 = skor_1 + 1
        return skor_1
    else:
        skor_1 = skor_1 + 5 - tanggungan
    return skor_1

def rataan(a,b,c):
    return (a+b+c) / 3

def token_lstrk(rataan):
    skor_2 = 0
    if rataan > 100000:
        skor_2 = skor_2 + 3
        return skor_2
    elif rataan > 50000 and rataan <=100000:
        skor_2 = skor_2 + 2
        return skor_2
    else:
        skor_2 = skor_2 + 1
        return skor_2
```



```

def gaji(total_gaji):
    skor_3 = 0
    if total_gaji > 10000000:
        skor_3 = skor_3 + 7
        return skor_3
    elif total_gaji > 8000000 and total_gaji <= 10000000:
        skor_3 = skor_3 + 6
        return skor_3
    elif total_gaji > 6000000 and total_gaji <= 8000000:
        skor_3 = skor_3 + 5
        return skor_3
    elif total_gaji > 4000000 and total_gaji <= 6000000:
        skor_3 = skor_3 + 4
        return skor_3
    elif total_gaji > 3000000 and total_gaji <= 4000000:
        skor_3 = skor_3 + 3
        return skor_3
    else:
        skor_3 = skor_3 + 2
        return skor_3

def cek_KIP(KIP_K):
    skor_4 = 0
    if KIP_K == 'Ya':
        skor_4 = skor_4 + 1
        return skor_4
    else:
        skor_4 = skor_4 + 5
        return skor_4

def skor_tot(a,b,c,d):
    return (a*0.2)+(b*0.3)+(c*0.2)+(d*0.3)

```

```

uang_pokok = 750000
tanggungan = 3
bln1,bln2,bln3 = 120000,75000,50000
total_gaji = 5500000
KIP_K = 'Tidak'
print(jml_tanggungan(tanggungan))
print(token_lstrk(rataan(bln1,bln2,bln3)))
print(gaji(total_gaji))
print(cek_KIP(KIP_K))
skor = skor_tot(jml_tanggungan(tanggungan),token_lstrk(rataan(bln1,bln2,bln3)),
                gaji(total_gaji),cek_KIP(KIP_K))

```

[38] ✓ 0.4s

... 2
2
4
5

```

jml_ukt = uang_pokok + skor * 500000
jml_ukt

```

[39] ✓ 0.5s

... 2400000.0

Turunan Polinom

```
def splt(dat):
    return dat.replace(' ','').replace('-','+-').split('+')
def chdepan(dat):
    return dat[1:] if dat[0]!='' else dat
def eqkan(dat):
    return map( lambda x: x if '^' in x else x+ '^1' if 'x' in x else x+ 'x^0', dat)
def toarr2d(dat):
    return r( lambda a,b:a + [ [float(hurf) for hurf in b.split('x^')] ] ,dat,[] )
def sortdesc(dat):
    return sorted(dat,key=lambda x:x[1],reverse=True)
def culture(dat):
    return map( lambda x: [0,0] if x[1]==0 else [ x[1] * x[0] , x[1]-1 ], dat)
def tostr(dat):
    return map( lambda x: '0' if x[0]==0 else str(x[0]) if x[1]==0 else str(x[0] + 'x' if x[1]==1 else str(x[0]) + 'x^' + str(i)))
def prettykan(dat):
    return r( lambda a,b: a+'+' + b if b!='0' else a ,dat,'')
def prettysign(dat):
    return dat.replace('+-',' -').replace('+',' + ')
```

✓ 0.7s

+ Code

+ Markdown

```
dat = '-3x^5 + 2x^2 - 4x + 5'
fss = (splt,chdepan,eqkan,toarr2d,sortdesc,culture,tostr,prettykan,prettysign)
my_turunan = mycompose(*fss)
```

✓ 0.5s

splt(dat)

✓ 0.7s

['', '-3x^5 ', ' 2x^2 ', '- 4x ', ' 5']

Buatlah fungsi untuk menghitung biaya yang harus dibayar customer pada suatu e-commerce menggunakan higher order function. Buatlah decorator untuk mengeluarkan harga sebelum pajak dan sesudah pajak (pajak = 11%) ! Gunakan decorator untuk menambahkan perhitungan waktu eksekusi!

```
from functools import reduce as r
keranjang = [
    {'Jumlah Barang' : 5, 'Harga' : 10},
    {'Jumlah Barang' : 7, 'Harga' : 20},
    {'Jumlah Barang' : 20, 'Harga' : 4.5}
]

def pajak_decorator (func):
    def inner(*args , **kwargs):
        res = func(*args, **kwargs)
        print ('Sub Total:', res)
        print ('Pajak:', res * 0.11)
        print ('Total:', res + res*0.11)
        return res
    return inner

import time

def calc_time_decorator(fu):
    def inner (*args, **kwargs):
        waktu_awal = time.time()
        res = fu(*args, **kwargs)
        waktu_akhir = time.time()
        print ('Waktu Eksekusi:', waktu_akhir-waktu_awal)
        return res
    return inner
```

✓ 0.5s

```
@calc_time_decorator
```

```
@pajak_decorator
```

```
def hitung_pembayaran_1(keranjang):
```

```
    return r(lambda a,b: a+ (b['Jumlah Barang']*b['Harga']),keranjang,0) * 1000
```

```
hitung_pembayaran_1(keranjang)
```

✓ 0.8s

Sub Total: 280000.0

Pajak: 30800.0

Total: 310800.0

Waktu Eksekusi: 0.0

```
@calc_time_decorator
```

```
@pajak_decorator
```

```
def hitung_pembayaran_2(keranjang):
```

```
    s = 0
```

```
    for k in keranjang :
```

```
        s = s + k ['Jumlah Barang'] * k ['Harga']
```

```
    return s * 1000
```

```
hitung_pembayaran_2(keranjang)
```

✓ 0.5s

Sub Total: 50000

Pajak: 5500.0

Total: 55500.0

Waktu Eksekusi: 0.0