

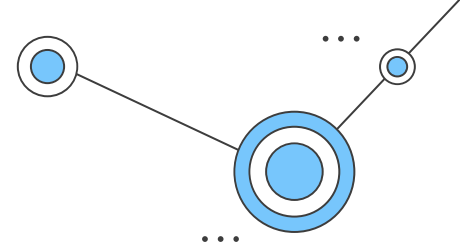


Laporan Akhir Praktikum

Pemrograman Berbasis Fungsi (RB)

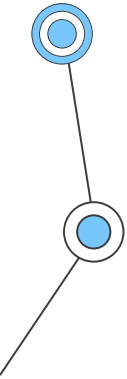
Fabio Hedfam G. Siregar
120450100
Sains Data

Contents of Report



Here's what you'll find in this Report :

1. Hasil dokumentasi dan pengerjaan program atau soal pada *ppt* pertemuan kelas mulai dari pertemuan 9-13.
2. Hasil dokumentasi dan pengerjaan program atau soal pada *pre-test*, Jurnal, dan *post-test* praktikum.





Latihan

Table of Contents (Latihan)

01

HOF (Filter)

The filter() function extracts elements from an iterable (list, tuple etc.) for which a function returns True.

02

HOF (Reduce)

Python's reduce() implements a mathematical technique commonly known as folding or reduction.

03

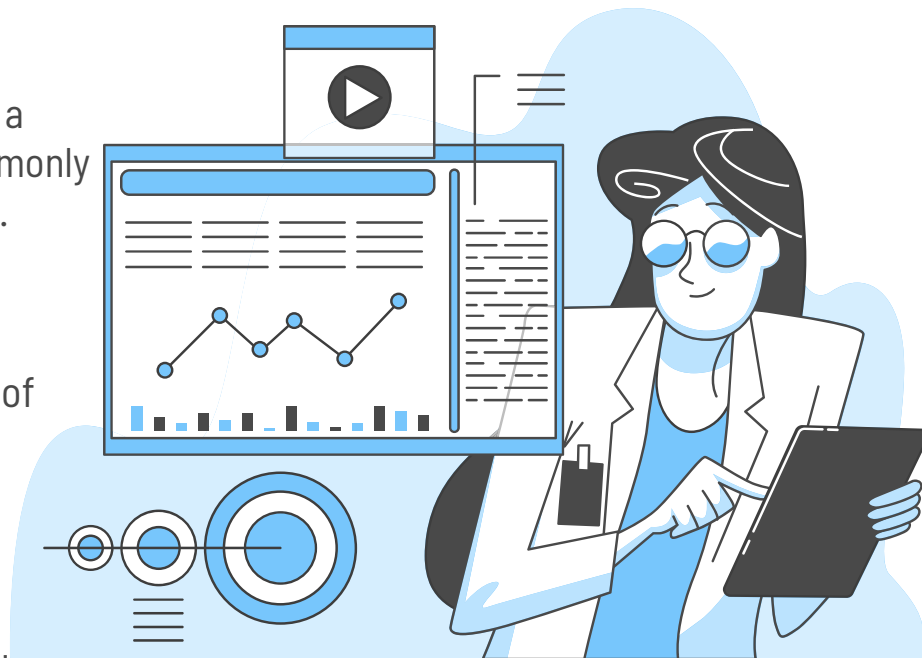
Recursion

Recursion is the process of defining something in terms of itself.

04

Purity and Immutability

a function that performs an operation whose result depends only on the function's input.





Function Building Function

Function composition is the way of combining two or more functions in such a way that the output of one function becomes the input of the second function and so on.

Table of Contents



01

HOF (Filter)

The filter() function extracts elements from an iterable (list, tuple etc.) for which a function returns True.

Understanding the Problem



Soal Pertama

Buat program untuk menghitung deret bilangan prima dari 2 hingga N menggunakan HOF filter dan map



Soal Kedua

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  
}
```





Documentation



01

Soal Pertama

Buat program untuk menghitung deret bilangan prima dari 2 hingga N menggunakan HOF filter dan map

Jawaban

```
def is_prime(n):  
    return True not in [n%k==0 for k in range(2,n)]  
  
def primes(m):  
    return [n for n in range(2,m) if is_prime(n)]
```

```
print(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]
```




Documentation



02

Soal Kedua

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  
}
```

Jawaban

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

```
cnt_emp = lambda lin, employee: r( lambda a,b: a+1 if b[1]>lin else a, employee.items(),0)  
cnt_emp(25,employee)
```

02

HOF(Reduce)

Python's `reduce()` implements a mathematical technique commonly known as folding or reduction.



Documentation



01

Soal Pertama

Buat fungsi mencari jumlah
bilangan genap dari list L!

Jawaban

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

```
r (lambda a,b:a+(1 if b % 2 == 0 else 0), L,0)
```



Documentation



02

Soal Kedua

Buat fungsi untuk menghitung n!
Menggunakan reduce!

Jawaban

```
n = 4  
print(r(lambda x,y: x*y, range (1,n+1)))
```



Documentation



03

Soal Ketiga

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

Jawaban

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

```
euclidian = lambda X,Y: r(lambda a,c:a+c, map(lambda x,y: (x-y)**2, X,Y))**0.5  
euclidian(X,Y)
```

10.583005244258363

Documentation

03

Soal Ketiga

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  
}
```

Jawaban

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

```
cnt_emp = lambda lin, employee: r( lambda a,b: a+1 if b[1]>lin else a, employee.items(),0)  
cnt_emp(25,employee)
```



Documentation



04

Soal Keempat

Buatlah deret fibonacci menggunakan higher order function!

Jawaban

```
fib = lambda n: r(lambda x, _: x+[x[-1]+x[-2]], range(n-2), [0, 1])  
fib(10)
```

✓ 0.1s

[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

03

Recursion

Recursion is the process of defining something in terms of itself.



Documentation



01

Soal

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

Jawaban non-rekursif

```
def fibonacci(n):  
    if n == 1:  
        return [1]  
    if n == 2:  
        return [1, 1]  
    fibs = [1, 1]  
    for _ in range(2, n):  
        fibs.append(fibs[-1] + fibs[-2])  
    return fibs  
  
print(fibonacci(500))
```

```
86168291600238450732788312165664788095941068326060883324529903470149056115823592713458328176574447204501,  
139423224561697880139724382870407283950070256587697307264108962948325571622863290691557658876222521294125]
```



Documentation



01

Soal

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

Jawaban Rekursif

```
def fib(n): ...  
    if (n <= 2):  
        return 1  
    else:  
        return (fib(n-1)+fib(n-2))  
print ("fungsi untuk menampilkan deret fibonacci sebanyak x buah")  
n = 500  
for i in range (1,n):  
    print (fib(i))
```


```
86168291600238450732788312165664788095941068326060883324529903470149056115823592713458328176574447204501,  
139423224561697880139724382870407283950070256587697307264108962948325571622863290691557658876222521294125]
```



04

Purity and Immutability

a function that performs an operation whose result depends only on the function's input.



Documentation

01

Soal

Ubah fungsiku menjadi pure function!

```
def fungsiku(L):  
    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
```

Jawaban

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

[16, 3.0, 22, 4.0]

Documentation

02

Soal

Ubah fungsiku2 menjadi pure function!

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

Jawaban

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

[16, 3.0, 22, 25]

[16, 3.0, 22, 25]



Documentation



03

Soal

Apakah isi dalam tuple tup ada yang dapat diubah?

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

Jawaban

Tidak dapat diubah

05

Function Building Function

Function composition is the way of combining two or more functions in such a way that the output of one function becomes the input of the second function and so on.



Documentation



01

Soal

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. Mengurangi input dengan 2 kali input nya
 3. Mengeluarkan nilai kuadrat dari input nya

B. Buatlah fungsi invers nya!



01

Jawaban

Documentation



```
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))
ny_f_kom = f_komp(kurku, f_komp(powku, addku))

ny_f_kom(10)
```

✓ 0.3s

-400

```
# 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(-400)
```

✓ 0.1s

10.0



Documentation



01

Jawaban

```
def compose(*funcs):  
    def inner(f,g):  
        return lambda x:f(g(x))  
    return r(inner, reversed(funcs), lambda x:x)
```

✓ 0.1s

```
mycomp = compose(addku, powku, kurku)  
mycomp(10)
```

✓ 0.1s

-400



Documentation



02

Soal

Latihan Penentuan UKT Mahasiswa

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



Documentation



02

Soal

Latihan : Penentuan UKT Mahasiswa

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



Documentation



02

Soal

Latihan : Penentuan UKT Mahasiswa

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



Documentation



02

Soal

Latihan : Penentuan UKT Mahasiswa

Perhitungan pembayaran UKT adalah sebagai berikut:

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

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

$$\text{Uang Pokok} = 750 \text{ ribu}$$



Documentation



02

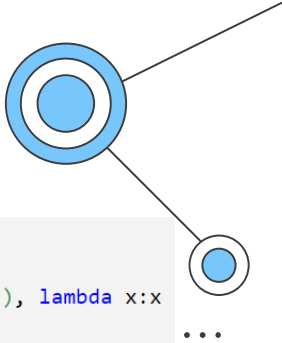
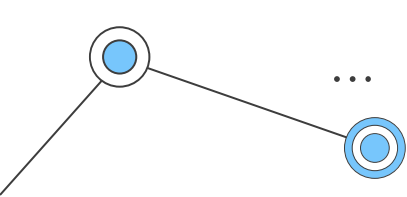
Soal

Latihan : Penentuan UKT Mahasiswa

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



Documentation

```
from functools import reduce as r
# Define function composition
mycompose = lambda *funcs: r( lambda f, g: lambda x: f(g(x)), reversed(funcs), lambda x:x
```

✓ 0.1s

Jawaban

02

```
# Ketentuan jumlah tanggungan
def skor1(jtg):
    return 1 if jtg >= 5 else 5-jtg
```

✓ 0.1s

```
# Ketentuan token listrik
def skor2(X):
    def rata(X):
        return sum(X)/len(X)

    def l_cond_1(X):
        return [X, [X>100000] ]

    def l_cond_2(X):
        return [X[0], X[1] + [ X[0] >= 50000 ] ]

    def to_score2(X):
        return r( lambda a,b: a+ (1 if b == True else 0), X[1], 1)

    compose_cond = mycompose(rata, l_cond_1, l_cond_2, to_score2)
    return compose_cond(X)
```


Documentation

02

Jawaban

```
# Ketentuan gaji
def con_1(X):
    return [X[0], 1, X[2], [ X[0] > X[2][X[1]] ] ]
def con_2_to_n(X):
    return [X[0], X[1]+1, X[2], X[3] + [ X[0] > X[2][X[1]] ] ]
def to_score(X):
    return r( lambda a,b: a+ (1 if b == True else 0), X[-1], 2)
def prep(gj):
    return [gj, 0, list(map( lambda x: x*100000, list(range(10,3,-1)) + [3]) )]
def skor3(gaji):
    comppy = mycompose(prepare, con_1, *(con_2_to_n for i in range(4)), to_score)
    return comppy(gaji)
```

✓ 0.1s

```
# Ketentuan KIP K
def skor4(X=True):
    return 1 if X else 5
```



Documentation



02

Jawaban

```
def combineskor(X):  
    return X + [map( lambda f,x: f(x), X[1], X[0] )]  
def boboti(X):  
    return r( lambda a,b: a+b, map(lambda x,y: x*y, X[-1], [0.2, 0.3, 0.2, 0.3]) )  
def toUKT(X):  
    return 750000 + X*500000
```

✓ 0.9s

```
mhs = [3,  
    [120000, 75000, 50000],  
    5.5 * 10**6,  
    False  
]  
datas = [mhs, [skor1, skor2, skor3, skor4] ]  
compose_fin = mycompose(combineskor, boboti, toUKT)  
compose_fin(datas)
```



Documentation



03

Soal

Latihan: Turunan Polinom

Contoh input:

```
dat = '-3x^5 + 2x^2 - 4x + 5'
```

Output:

```
' -15.0x^4 + 4.0x -4.0'
```

Documentation

03

Jawaban

```
# Turnan polinom
```

```
def split(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 calctur(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^' + str(x[1]), dat)  
def prettykan(dat):  
    return r( lambda a,b: a+'+' + b if b != '0' else a, dat, '')  
def prettysign(dat):  
    return dat.replace('+-', ' -').replace('+', '+ ')
```



Documentation



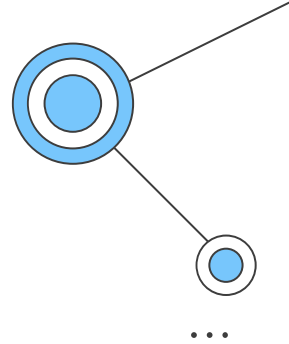
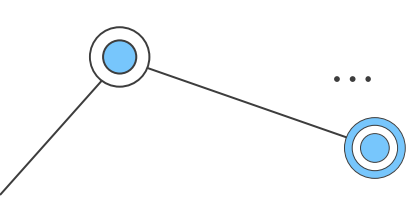
03

Jawaban

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

✓ 0.9s

```
' -15.0x^4.0+ 4.0x^1.0 -4.0'
```



Documentation

04

Soal

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!

Contoh input:

```
keranjang = [  
    {'Jumlah_Barang': 5 , 'Harga': 10 },  
    {'Jumlah_Barang': 7 , 'Harga': 20 },  
    {'Jumlah_Barang': 20 , 'Harga': 4.5 }  
]
```



04

Jawaban

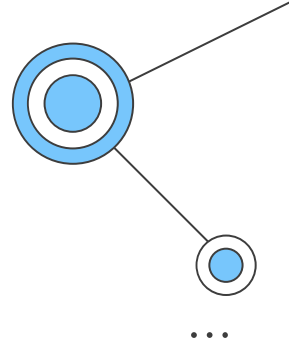
Documentation

```
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 * 0.01)
        print('Total: ', res + res * 0.01)
        return res
    return inner

import time

def calc_time_decorator(func):
    def inner(*args, **kwargs):
        start = time.time()
        res = func(*args, **kwargs)
        end = time.time()
        print('Time: ', end - start)
        return res
    return inner
```



Documentation

04

Jawaban

```
@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)
```

] ✓ 0.9s

Sub Total: 280000.0

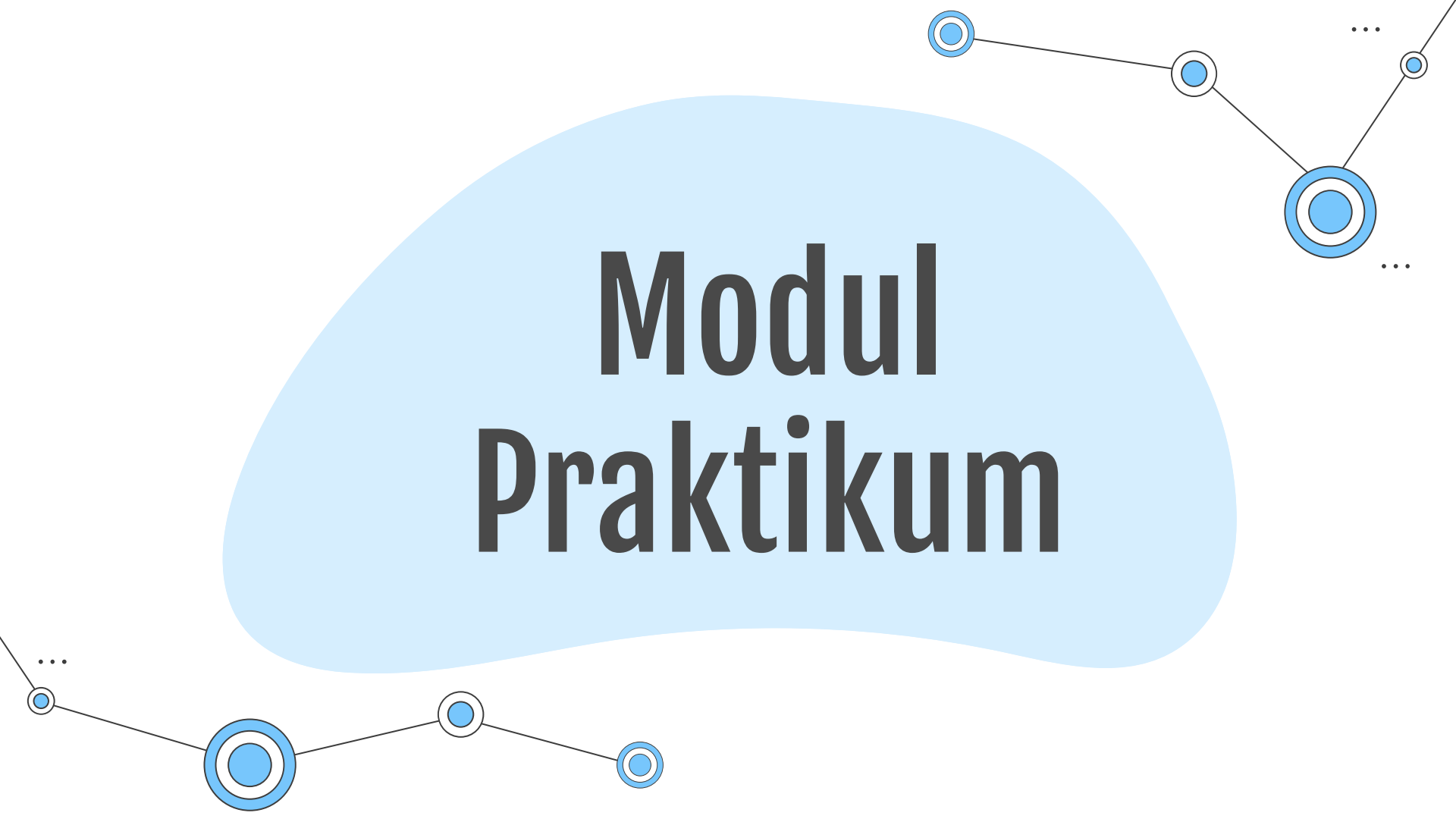
Pajak: 2800.0

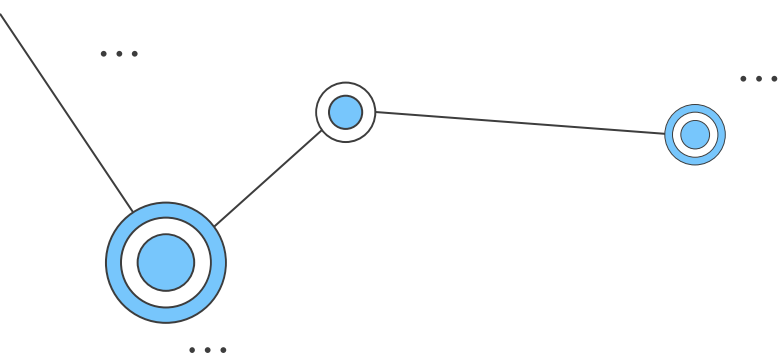
Total: 282800.0

Time: 0.0

280000.0

Modul Praktikum





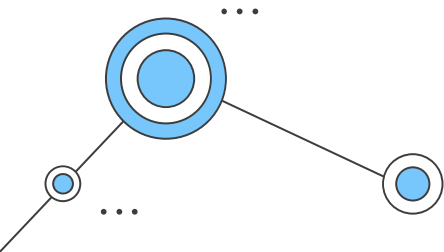
Tujuan Praktikum :

- Praktikan mampu memodularisasikan algoritma dalam python
- Praktikan mampu menerapkan fungsi anonymous dalam python

Modul 1

Tugas Praktikum :

- Tugas Pendahuluan Praktikum
- Jurnal Praktikum



TASK

TP

Seorang mahasiswa telah menyewa beberapa buku dari perpustakaan kota. Maksimal jumlah hari sewa buku adalah 26 hari. Buku yang telah disewa adalah sebagai berikut:

1. PBF, 100 halaman (Rp 1.000 per hari)
2. ALSTRAT, 250 halaman (Rp 1.000 per hari)
3. SWARM, 200 halaman (Rp 1.500 per hari)
4. BASDAT, 350 halaman (Rp 1.500 per hari)
5. SSD, 400 halaman (Rp 2.000 per hari)

Buku tersebut akan difotokopi oleh mahasiswa sebagai referensi bahan belajar. Mesin fotokopi hanya mampu memfotokopi sebanyak 50 halaman dalam 1 hari. Setelah semua halaman buku selesai difotokopi, buku-buku tersebut akan dikembalikan ke perpustakaan. Hitunglah total biaya sewa yang harus dibayarkan oleh mahasiswa tersebut dan perkirakan berapa hari yang dibutuhkan untuk pengembaliannya.

Buatlah program dengan memodularisasikan algoritma ke dalam fungsi-fungsi untuk permasalahan tersebut. **Tidak diperkenankan menggunakan library apapun**

Jurnal

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

- a. Harga sewa buku berbeda-beda sesuai dengan kategorinya
- b. Harga sewa buku dihitung berdasarkan jumlah halaman nya
- c. Harga sewa buku dihitung per hari nya
- d. 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!



Tugas Pendahuluan Jawaban

```
buku = ['PBF', 'ALSTRAT', 'SWARM', 'BASDAT', 'SSD']  
max_print = 50
```

```
def hal_buk(hal):  
    tot_hal = {"PBF": 100, "ALSTRAT": 250, "SWARM": 200, "BASDAT": 350, "SSD": 400}  
    halaman = tot_hal[hal]  
    return halaman
```

```
def har_buk(x):  
    harga = {"PBF": 1000, "ALSTRAT": 1000, "SWARM": 1500, "BASDAT": 1500, "SSD": 2000}  
    hargab = harga[x]  
    return hargab
```

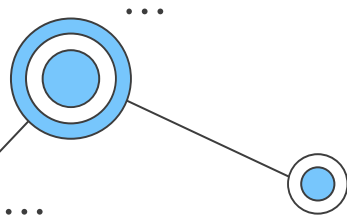
```
jum_hal = list(map(hal_buk, buku))  
harga_buku = list(map(har_buk, buku))
```

```
har_pin = lambda x: sum(x) / max_print  
halbuk = round(har_pin(jum_hal))  
print(f' Jumlah hari pinjam terhitung {halbuk} hari ')
```

Jumlah hari pinjam terhitung 26 hari

```
tot_har = lambda x: halbuk * sum(x)  
print(f' berikut jumlah yang harus dibayar: Rp ', tot_har(harga_buku))
```

berikut jumlah yang harus dibayar: Rp 182000

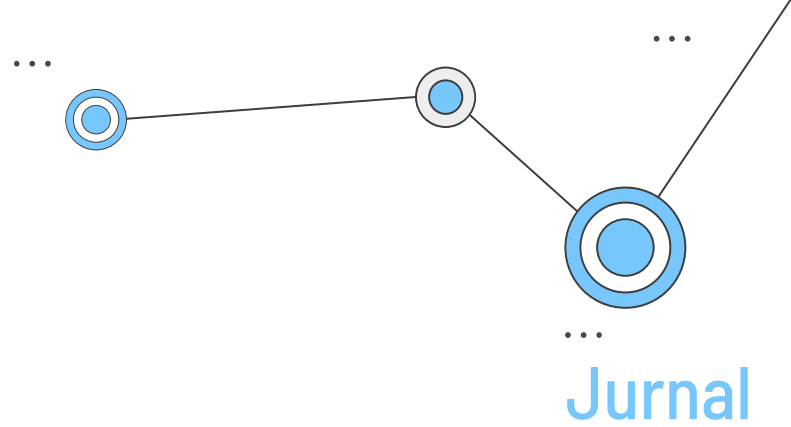


Tanggal : 2022-05-17

Durasi = 10

```
tanggal = input('Tanggal Pinjam: ')\ndurasi = int(input('Durasi Pinjam (hari): '))
```

```
kategori = {\n    1: 100,\n    2: 200,\n    3: 250,\n    4: 300,\n    5: 500,\n}
```



...



...



...

Jurnal

```
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))]
```

```
tgl_p = dtl(tanggal)
wkt_kembali(tgl_p,durasi)
```

✓ 0.6s

[2022, 5, 27]

```
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
```

✓ 0.5s

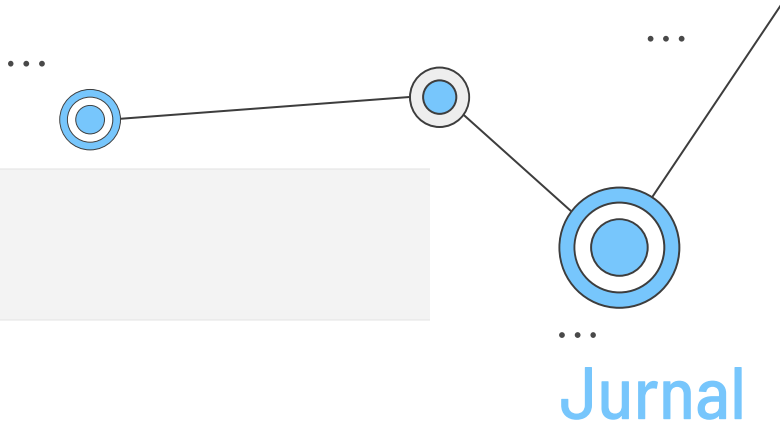
+ Code

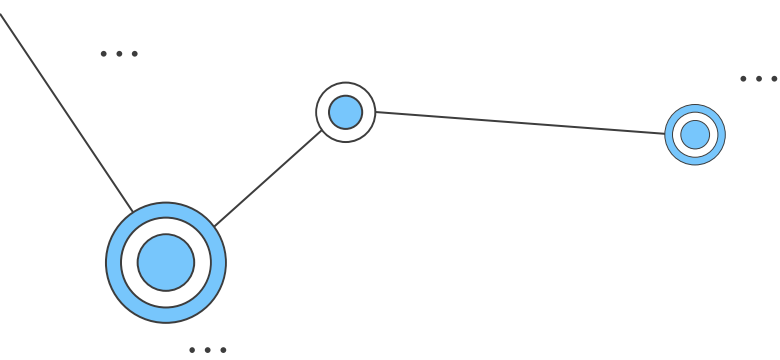
+ Markdown

```
calc_all_biaya(kategoris, sewaan_all, durasi)
```

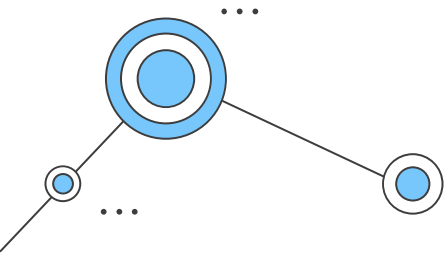
✓ 0.3s

24000





Modul 2



Tujuan Praktikum :

- Praktikan mampu memahami fungsi HOF dalam python
- Praktikan mampu menerapkan fungsi HOF dalam python

Tugas Praktikum :

- Tugas Pendahuluan Praktikum
- Jurnal Praktikum

TASK

TP

Jurnal

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

1. Buatlah sebuah fungsi bernama `ulangi_NIM`, `ulangi` memiliki input sebuah bilangan skalar `a`, dan mengeluarkan vektor `1xn` dengan seluruh elemen nya adalah `a` !
2. Buatlah deret bilangan sebagai berikut dengan input `n` sebagai panjang deret:

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

3. Jumlahkan deret bilangan tersebut!
4. Sebuah DNA dimodelkan dalam sebuah string menjadi sequence TCGA dan disimpan ke dalam data :

<https://drive.google.com/file/d/18C1ylsTXrY9pglqqlhijoS8LYmcxdIjM/view?usp=sharing>

hitunglah kemunculan pola ACT pada data tersebut!

TASK

Jurnal

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

1. Buatlah sebuah fungsi bernama ulangi_NIM, ulangi memiliki input sebuah bilangan skalar a, dan mengeluarkan vektor 1xn dengan seluruh elemen nya adalah a !
2. Buatlah deret bilangan sebagai berikut dengan input n sebagai panjang deret:

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

3. Jumlahkan deret bilangan tersebut!
4. Sebuah DNA dimodelkan dalam sebuah string menjadi sequence TCGA dan disimpan ke dalam data :

<https://drive.google.com/file/d/18C1ylsTXrY9pglqqlhijoS8LYmcxdIjM/view?usp=sharing>

hitunglah jumlah kemunculan pola berikut pada data tersebut:

- a. A
- b. AT
- c. GGT
- d. AAGC
- e. AGCTA

TASK

Jurnal

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

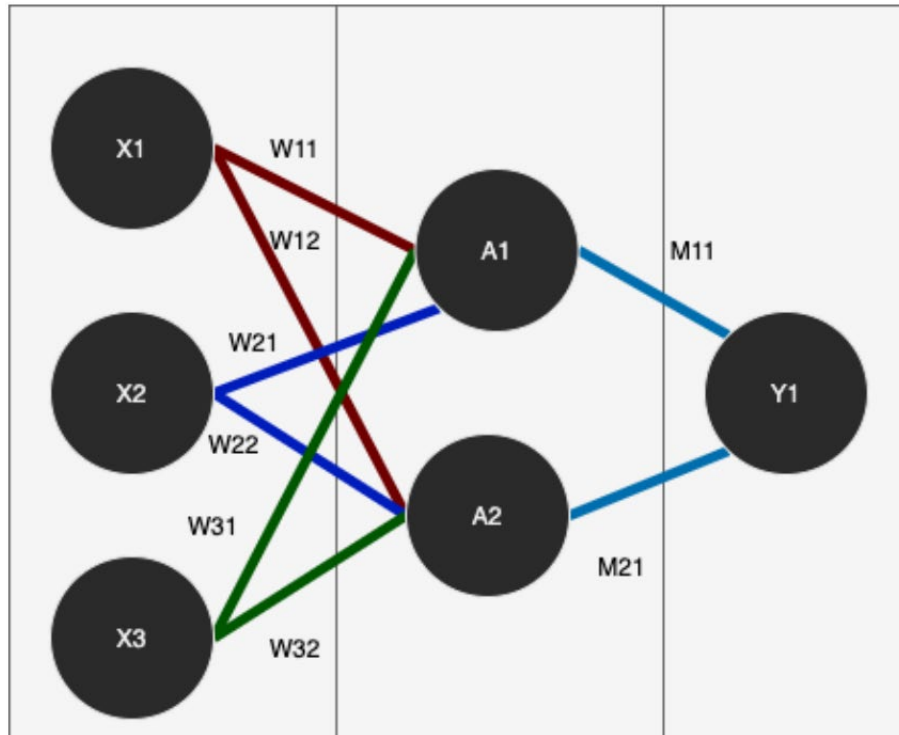
Reverse complmenet : TGACT

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

TASK

Jurnal

6. Perhatikan Neural Network dibawah ini:



TASK

Jurnal

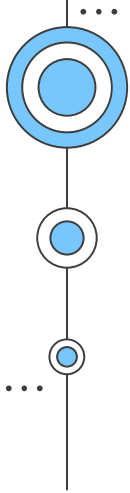
Terdapat proses yang dinamakan feed-forward. Input dalam sebuah neural network diproses ke hidden layer hingga ke output layer.

Setiap Node, menunjukan neuron dan setiap garis menunjukan 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:

$$S_1 = X_1.W_{11} + X_2.W_{21} + X_3.W_{31}$$



Jurnal

TASK

$$S_2 = X_2 \cdot W_{12} + X_2 \cdot W_{22} + X_3 \cdot W_{32}$$

$$A_1 = \frac{1}{1 + e^{-S_1}}$$

$$A_2 = \frac{1}{1 + e^{-S_2}}$$

$$Z_1 = M_{11} \cdot A_1 + M_{21} \cdot A_2$$

$$Y_1 = \frac{1}{1 + e^{-Z_1}}$$

Buatlah fungsi feed-forward dengan input berikut:

$$W_{11} = 0.5$$

$$W_{12} = 0.4$$

$$W_{21} = 0.3$$

$$W_{22} = 0.7$$

$$W_{31} = 0.25$$

$$W_{32} = 0.9$$

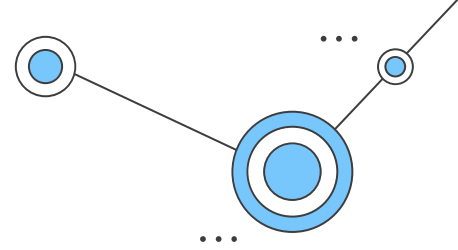
$$M_{11} = 0.34$$

$$M_{21} = 0.45$$

dan $X_1 = 9$, $X_2 = 10$, $X_3 = -4$



Tugas Pendahuluan



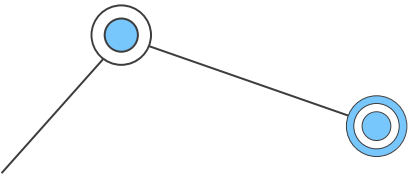
Jawaban

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

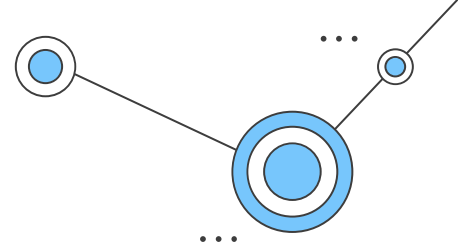
```
x=int(input('enter the sum of the length of the series : '))
a=list(range(1,x+1))

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

[0.5, -0.25, 0.125, -0.0625, 0.03125]



Tugas Pendahuluan



Jawaban

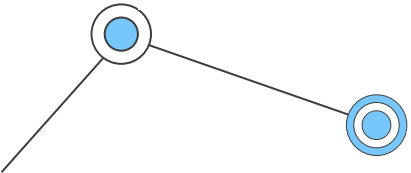
Jumlahkan deret bilangan tersebut!

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

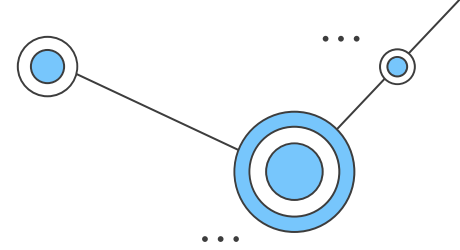
a: 0.5 , b: -0.25 -> a+b: 0.25

a: 0.25 , b: 0.125 -> a+b: 0.375

0.375



Tugas Pendahuluan



Jawaban

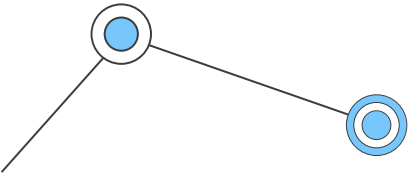
Sebuah DNA dimodelkan dalam sebuah string menjadi sequence TCGA dan disimpan ke dalam data :

```
f = open("dna.txt", "r")
print(f.read(5))
```

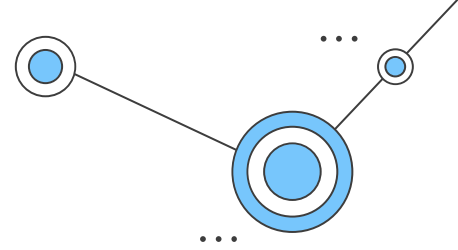
TGTCT

```
def count_matches(dna, pattern):
    indexes = range(0, len(dna)-len(pattern))
    matches_start_indexes = filter(lambda start_index: dna[start_index:start_index+len(pattern)] == pattern, indexes)
    return len(list(matches_start_indexes))
```

✓ 0.7s



Tugas Pendahuluan



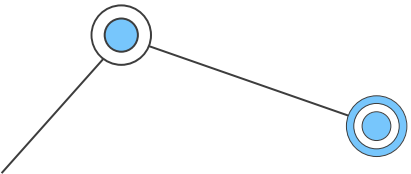
Jawaban

Buatlah sebuah fungsi bernama `ulangi_NIM`, `ulangi` memiliki input sebuah bilangan skalar `a`, dan mengeluarkan vektor $1 \times n$ dengan seluruh elemen nya adalah `a` !

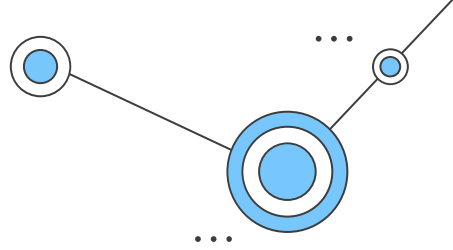
```
a = int(input("Masukkan jumlah bilangan skalar a : "))
L = list(map(int,input("\nMasukkan bilangan skalar a : ").strip().split()))[:a]
def ulangi_100(n):
    return n * 1
print(list(map(ulangi_100, L)))
```

Py

[2]



Jurnal



Jawaban

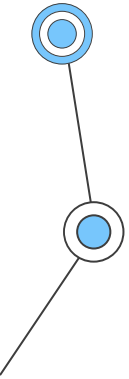
Buatlah sebuah fungsi bernama `ulangi_NIM`, `ulangi` memiliki input sebuah bilangan skalar `a`, dan mengeluarkan vektor $1 \times n$ dengan seluruh elemen nya adalah `a` !

```
n = 8
def ulangi_100(n):
    a = list(map(lambda n: 1*n, range(1,n+1))) #scalar number code
    return a
print('bilangan skalar :', n)
ulangi_100(n)
```

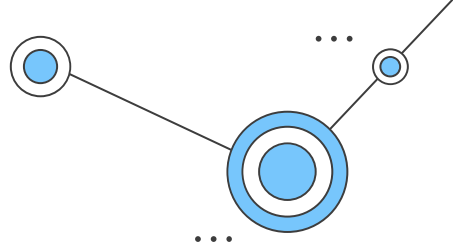
P2

bilangan skalar : 8

[1. 2. 3. 4. 5. 6. 7. 8]



Jurnal



Jawaban

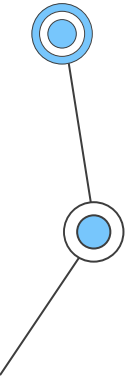
Buatlah sebuah fungsi bernama `ulangi_NIM`, `ulangi` memiliki input sebuah bilangan skalar `a`, dan mengeluarkan vektor $1 \times n$ dengan seluruh elemen nya adalah `a` !

```
n = 8
def ulangi_100(n):
    a = list(map(lambda n: 1*n, range(1,n+1))) #scalar number code
    return a
print('bilangan skalar :', n)
ulangi_100(n)
```

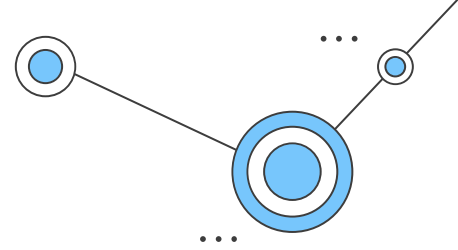
P2

bilangan skalar : 8

[1. 2. 3. 4. 5. 6. 7. 8]



Jurnal



Jawaban

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

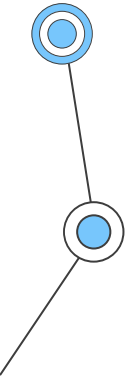
```
#row length
n = 5
deret = list(map(lambda x: ((-1)*(x+1)) * (1/(2*x)), range(1,n+1)))
print(deret)
```

```
[-1.0, -0.75, -0.6666666666666666, -0.625, -0.6000000000000001]
```

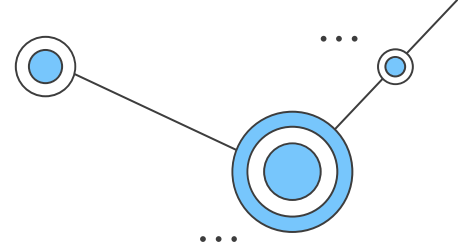
```
b = range(1,n+1)

def pola_deret(x):
    return ((-1)*(x)) * (1/(2*(x+1)))
print(list(map(pola_deret, b)))
```

```
[-0.25, -0.3333333333333333, -0.375, -0.4, -0.4166666666666663]
```



Jurnal



Jawaban

Jumlahkan deret bilangan tersebut!

- first, import library "functools" to retrieve reduce . function
- second, write the function with reduce that we have

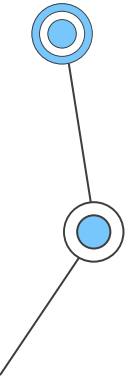
```
from functools import reduce
print(reduce(lambda x,y: x+y, deret))
```

-3.6416666666666666

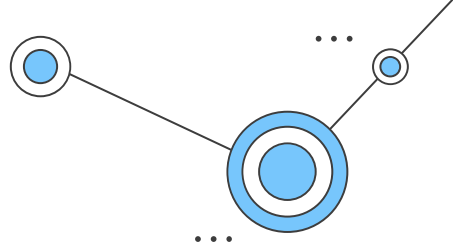
Sebuah DNA dimodelkan dalam sebuah string menjadi sequence TCGA dan disimpan ke dalam data :

the first dat is used to read "txt", then the second dat is used to remove "/n" in the data when it is read

```
dat = open("dna.txt", 'r').read()
dat=dat[:-1]
seq='ACT'
```



Jurnal



Jawaban

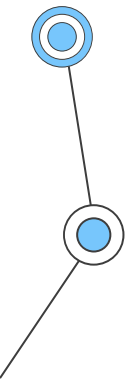
The append function is used to add n characters to i ; remap function is useful to remap all seq functions

```
seq="ACT"
def append_n(dat, i, n):
    return reduce(lambda x, y: x + y, [dat[i:i+n]])
append_n(dat, 0, 3)
```

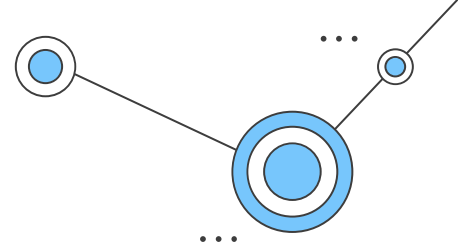
'TGT'

```
def remap(dat, seq):
    return map(lambda x:append_n(dat, x,len(seq)), range(0, len(dat)-len(seq)+1))
list(remap(dat,"ACT"))
```

```
def count_mer (dat,seq):
    return reduce(lambda x,y:x +(1 if y==seq else 0), remap(dat,seq), 0)
count_mer(dat,"ACT")
```



Jurnal



Jawaban

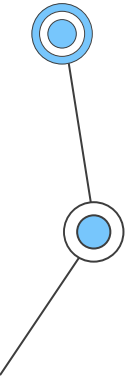
```
list(remap(dat, 'ACT'))[-1]  
len(dat)
```

6930

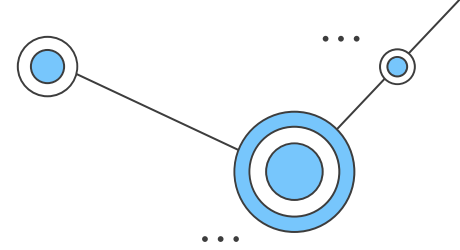
- *sequence is used for dictionary to store the result of count_mer*
- *count_all is used to count all seq*
- *res is used to call all number of seq we are looking for*

```
sequences=["A", "AT", "GGT", "AAGC", "AGCTA"]  
  
def count_all(dat, sequences):  
    return map(lambda x: count_mer(dat,x), sequences)  
  
res=count_all(dat,sequences)  
print(*res)
```

2112 557 77 22 5



Jurnal



Jawaban

- *complement is used as a library for complement*
- *reverse complement is used to convert complement to reverse*

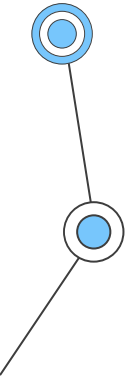
```
def komplemen(x):  
    return{'A':'T','T':'A','C':'G', 'G' : 'C'}.get(x)  
  
def reverse_komplemen(f):  
    return map(lambda x:komplemen(x),f)
```

✓ 0.4s

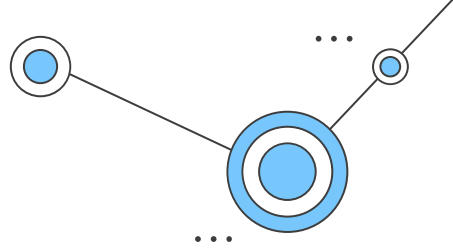
```
res = reverse_komplemen(dat) #to call all komplemen  
print(*res)
```

✓ 0.7s

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



Jurnal



Jawaban

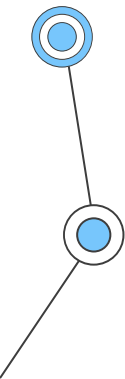
Number 6

- *first we need to call the math library*
- *def aktivasi is used to calculate activation function*
- *def WT_i is used to transpose matrix*
- *def WT is used to accomodate calculations*
- *def XW is used for calculations for one input*
- *def input_to_hidden is used to run the activation function*
- *W shape must be the same as M to match the pattern that has been made*

✓

```
import math

def aktivasi(x):
    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]))))
def XW(X,W):
    return map(lambda w: reduce(lambda a,b:a+b, map(lambda xx,ww:xx*ww,X,w),0),WT(W))
def input_to_hidden(X,W):
    return list(map(lambda x:aktivasi(x),XW(X,W)))
def feed_forward(X,W,M):
    return input_to_hidden(input_to_hidden(X,W),M)
```



Jurnal

Jawaban

```
X=[9,10,-4]
```

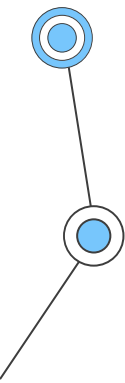
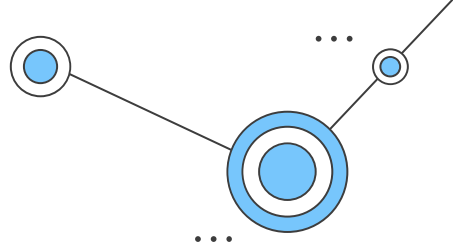
```
W=[[0.5,0.4],[0.3,0.7],[0.25,0.9]]
```

```
M=[[0.34],[0.45]]
```

```
feed_forward(X,W,M)
```

✓ 0.4s

```
[0.6876336740661236]
```



Terima Kasih

Pak/Bu atas ilmunya    