

Nama : Annisa Nugraheni

NIM : L200180066

Kelas : C

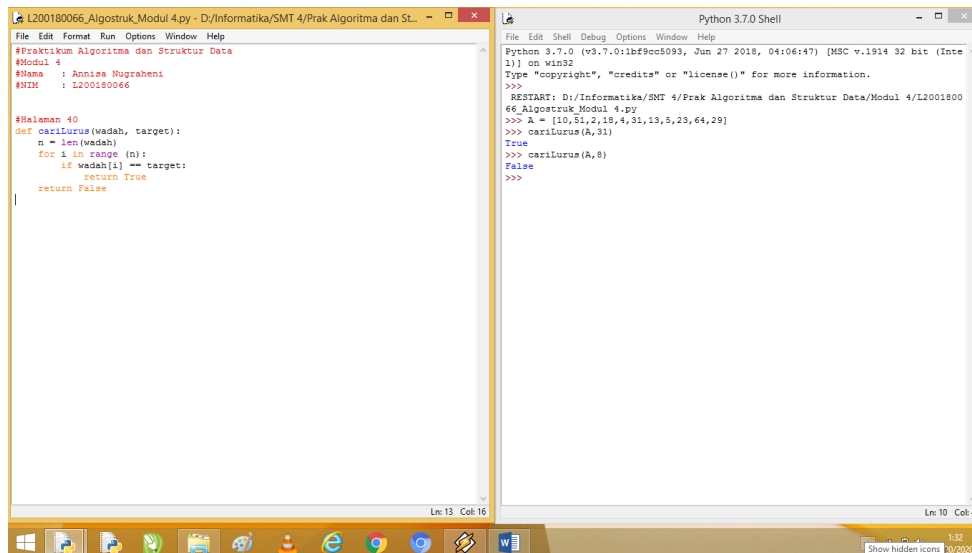
PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

MODUL 4

PENCARIAN

❖ 4.1. Linear Search

Berikut adalah screenshot program dan hasil dari latihan halaman 40:



The screenshot shows a Python IDE with two windows. The left window displays a Python script for a linear search function. The right window shows the execution results in a Python 3.7.0 Shell.

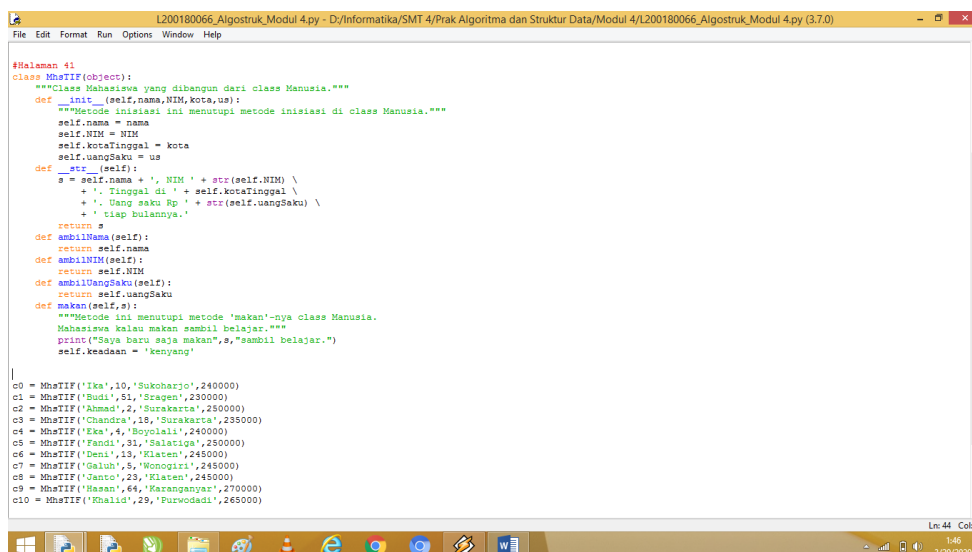
```
#Praktikum Algoritma dan Struktur Data
#Modul 4
#Nama : Annisa Nugraheni
#NIM : L200180066

#Halaman 40
def cariLurus(wadah, target):
    n = len(wadah)
    for i in range(n):
        if wadah[i] == target:
            return True
    return False
```

Python 3.7.0 Shell

```
Python 3.7.0 (v3.7.0:1bfecf5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 4/L200180066_Algostruk_Modul 4.py
>>> A = [10,51,2,18,4,31,13,5,23,64,29]
>>> cariLurus(A,31)
True
>>> cariLurus(A,8)
False
>>>
```

Berikut adalah screenshot program dari latihan halaman 41:



The screenshot shows a Python IDE with a single window displaying a Python script for a class-based program. The script defines a class named 'Mahasiswa' and creates several instances of it.

```
#Halaman 41
class Mahasiswa(object):
    """Class Mahasiswa yang dibangun dari class Manusia."""
    def __init__(self, nama, NIM, kota, us):
        """Metode inisiasi ini menutupi metode inisiasi di class Manusia."""
        self.nama = nama
        self.NIM = NIM
        self.kotaTinggal = kota
        self.uangSaku = us
    def __str__(self):
        s = self.nama + ', NIM ' + str(self.NIM) + \
            '\n . Tinggal di ' + self.kotaTinggal + \
            '\n . Uang saku Rp ' + str(self.uangSaku) + \
            '\n tiap bulannya.'
        return s
    def ambilNama(self):
        return self.nama
    def ambilNIM(self):
        return self.NIM
    def ambilUangSaku(self):
        return self.uangSaku
    def makan(self, s):
        """Metode ini menutupi metode 'makan'-nya class Manusia.
        Mahasiswa kalau makan sambil belajar."""
        print("Saya baru saja makan", s, "sambil belajar.")
        self.keadaan = 'kenyang'

c0 = Mahasiswa('Ika',10,'Sukoharjo',240000)
c1 = Mahasiswa('Budi',51,'Sragen',290000)
c2 = Mahasiswa('Ahmad',2,'Surakarta',290000)
c3 = Mahasiswa('Chandra',18,'Surakarta',235000)
c4 = Mahasiswa('Eka',4,'Boyolali',240000)
c5 = Mahasiswa('Fandi',31,'Salatiga',250000)
c6 = Mahasiswa('Deni',13,'Klaten',245000)
c7 = Mahasiswa('Galuh',5,'Wonogiri',245000)
c8 = Mahasiswa('Gento',23,'Klaten',245000)
c9 = Mahasiswa('Hanan',44,'Karanganyar',270000)
c10 = Mahasiswa('Khalid',29,'Purwodadi',265000)
```

```
L200180066_Algostruk_Modul 4.py - D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 4/L200180066_Algostruk_Modul 4.py (3.7.0)
File Edit Format Run Options Window Help
class Mahasiswa:
    def __init__(self):
        self.uangSaku = 0
    def __str__(self):
        s = self.nama + ', NIM ' + str(self.NIM) \
            + '\n Tinggal di ' + self.kotaTinggal \
            + '\n Uang saku Rp ' + str(self.uangSaku) \
            + '\n tiap bulannya.'
        return s
    def ambilNama(self):
        return self.nama
    def ambilNIM(self):
        return self.NIM
    def ambilUangSaku(self):
        return self.uangSaku
    def makan(self, s):
        """Metode ini menutupi metode 'makan'-nya class Manusia.
        Mahasiswa kalau makan sambil belajar."""
        print("Saya baru saja makan", s, "sambil belajar.")
        self.keadaan = 'kenyang'

c0 = Mahasiswa('Ika', 10, 'Sukoharjo', 240000)
c1 = Mahasiswa('Budi', 51, 'Sragen', 230000)
c2 = Mahasiswa('Ahmad', 2, 'Surakarta', 250000)
c3 = Mahasiswa('Chandra', 18, 'Surakarta', 230000)
c4 = Mahasiswa('Eka', 4, 'Boyolali', 240000)
c5 = Mahasiswa('Fandi', 31, 'Salatiga', 250000)
c6 = Mahasiswa('Dani', 13, 'Klaten', 245000)
c7 = Mahasiswa('Galuh', 5, 'Wonopring', 245000)
c8 = Mahasiswa('Janto', 23, 'Klaten', 245000)
c9 = Mahasiswa('Hasan', 64, 'Karanganyar', 270000)
c10 = Mahasiswa('Khalid', 28, 'Purwodadi', 265000)

#Lalu kita membuat daftar mahasiswa dalam bentuk list seperti ini:
Daftar = [c0, c1, c2, c3, c4, c5, c6, c7, c8, c9, c10]

target = 'Klaten'
for i in Daftar:
    if i.kotaTinggal == target:
        print(i.nama + ' tinggal di ' + target)

Ln: 44 Col: 0
3/20/2020
```

Berikut adalah screenshot hasil dari latihan halaman 41:

```
>>>
RESTART: D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 4/L200180066_Algostruk_Modul 4.py
Dani tinggal di Klaten
Janto tinggal di Klaten
>>>
```

Berikut adalah screenshot program dan hasil dari latihan halaman 42:

```
#Halaman 42
kumpulan = [1,2,3,4,5,0]

def cariTerkecil(kumpulan):
    n = len(kumpulan)
    #Anggap item pertama adalah yang terkecil
    terkecil = kumpulan[0]
    #Tentukan apakah item lain lebih kecil
    for i in range(1,n):
        if kumpulan[i] < terkecil:
            terkecil = kumpulan[i]

    return terkecil #Kembalikan yang terkecil
print(cariTerkecil(kumpulan))
```

```
RESTART: D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 4/L2001800
66_Algostruk_Modul 4.py
0
>>>
```

Ln: 56 Col: 4

```
Bagaimana programnya jika kita ingin mencari mahasiswa(dari class Mahasiswa di atas) yang uang sakunya terkecil
def kecil(Daftar):
    minim = Daftar[0].uangSaku
    for i in Daftar:
        if i.uangSaku < minim:
            minim = i.uangSaku
        if i.uangSaku == minim:
            nama = i.nama
    return nama, minim
print(kecil(Daftar))
```

```
RESTART: D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 4/L2001800
66_Algostruk_Modul 4.py
('Budi', 230000)
>>>
```

```

Bagaimana jika yang terbesar
def besar(Daftar):
    maxm = Daftar[0].uangSaku
    for i in Daftar:
        if i.uangSaku > maxm:
            maxm = i.uangSaku
            nama = i.nama
    return nama, maxm
print(besar(Daftar))

```

```

RESTART: D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 4/L2001800
66_Algostruk_Modul 4.py
('Hasan', 270000)
>>>

```

```

Bagaimanakah programnya jika kita ingin mencari semua mahasiswa yang uang sakunya kurang dari 250 ribu
def kurang(Daftar):
    a = []
    for i in Daftar:
        if i.uangSaku < 250000:
            a.append(i.nama)
    return a
print(kurang(Daftar))

```

```

>>>
RESTART: D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 4/L2001800
66_Algostruk_Modul 4.py
['Ika', 'Budi', 'Chandra', 'Eka', 'Deni', 'Galuh', 'Janto']
>>>

```

```

Yang lebih dari 250 ribu
def lebih(Daftar):
    a = []
    for i in Daftar:
        if i.uangSaku >= 250000:
            a.append(i.nama)
    return a
print(lebih(Daftar))

```

Ln: 127 Col: 0

```

RESTART: D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 4/L2001800
66_Algostruk_Modul 4.py
['Ahmad', 'Fandi', 'Hasan', 'Khalid']
>>>

```

Berikut adalah screenshot program dan hasil dari latihan halaman 43:

```

#Halaman 43
def binSe(list, target):
    #mulai dari seluruh runtutan elemen
    low = 0
    high = len(list) - 1

    #cara berulang belah runtutan itu menjadi separuhnya
    # sampai targetnya ditemukan
    while low <= high:
        #temukan pertengahan runtut itu
        mid = (high + low) // 2
        #Apakah pertengahannya semua target?
        if list[mid] == target:
            return True
        #atau targetnya di sebelah kirinya?
        elif target < list[mid]:
            high = mid - 1
        #atau targetnya ada di sebelah kanannya?
        else:
            low = mid + 1
        #jika runtutnya tidak bisa dibelah lagi, berarti targetnya tidak ada
    return False

list = [2,3,5,6,6,6,8,9,9,10,11,12,13,13,14]
target = 6
print(binSe(list,target))
list = [2,3,5,6,6,6,8,9,9,10,11,12,13,13,14]
target = 7
print(binSe(list,target))

```

Ln: 159 Col: 25

```

RESTART: D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 4/L2001800
66_Algostruk_Modul 4.py
True
False
>>>

```

Ln: 60 Col: 4

```
##Dapatkah kamu mengubah programnya agar dia mengembalikan index-nya kalau targetnya ditemukan,  
##dan mengembalikan False kalau target tidak ditemukan
```

```
def binSe(list, target):  
    a=[]  
    low = 0  
    high = len(list) - 1  
    while (low<=high):  
        mid = (low+high)//2  
        if (list[mid] == target):  
            a.append(list.index(target))  
            i=list.index(target)-1  
            j = list.index(target) + 1  
            while target == list[i]:  
                a.append(i)  
                i-=1  
            while target == list[j]:  
                a.append(j)  
                j+=1  
            return a  
        elif (target<list[mid]):  
            high = mid - 1  
        else:  
            low = mid + 1  
    return False  
  
list = [2,3,3,3,4,4,4,4,4,5,6,6,6,8,9,9,10,11,12,13,13,14]  
target = 6  
print(binSe(list,target))  
list = [2,3,5,6,6,6,8,9,9,10,11,12,13,13,14]  
target = 7  
print(binSe(list,target))
```

Ln: 186 Col: 25

```
~/...  
RESTART: D:/Informatika/SMT 4/Prak Algoritma dan Struktur Data/Modul 4/L200180066_Algostruk_Modul 4.py  
[10, 11]  
False  
>>> |
```

Ln: 64 Col: 4