

3. Nomer 6-7

```

Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\KULI AH\Tugas Semester 4\algotrunkModul 4\NOMER 1-4_150.py =====
>>> cariAba1()
Masukkan Nama: Klaten
[6, 8]
>>> cariUsteker112()
['Budi']
>>> cariKurang25()
Traceback (most recent call last):
  File "c:\python38\lib\site-packages\urllib\robotparser.py", line 1, in <module>
    from urllib.parse import urlparse
NameError: name 'cariKurang25' is not defined
>>> cariKurang25()
['Eka', 'Budi', 'Chandra', 'Eka', 'Deni', 'Galuh', 'Janto']
>>>
===== RESTART: D:\KULI AH\Tugas Semester 4\algotrunkModul 4\NOMER 5_150.py =====
>>>
Data 9 ada dalam linked list
Data 22 tidak ada dalam linked list
>>>
===== RESTART: D:\KULI AH\Tugas Semester 4\algotrunkModul 4\NOMER 6-7_150.py =====
>>>
>>> binde(14)
'Target pada indeks 4'
>>> binde(6)
'Target pada indeks 2'
>>> binde2(6)
[3, 4, 5]
>>> binde2(13)
[12, 13]
>>> ]

NOMER 6-7_150.py - D:\KULI AH\Tugas Semester 4\algotrunkModul 4\NOMER 6-7_150.py (1.8.2)
File Edit Format Run Options Window Help
A = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22]
def binde(target):
    low = 0
    high = len(A)
    while low < high:
        mid = (high + low) // 2
        if A[mid] == target:
            return "Target pada indeks " + str(mid)
        elif target < A[mid]:
            high = mid - 1
        else:
            low = mid + 1
    return False
def binde2(target):
    low = 0
    high = len(B)
    x = []
    while low < high:
        if B[low] == target:
            x.append(low)
            low += 1
        else:
            low += 1
    return x
B = [2, 3, 5, 6, 6, 6, 8, 9, 9, 10, 11, 12, 13, 13, 14]
>>> binde2(target):
low = 0
high = len(B)
x = []
while low < high:
    if B[low] == target:
        x.append(low)
        low += 1
    else:
        low += 1
return x

```

4. Nomer 8

Karena menggunakan konsep Big-O. Dimana yang dipakai adalah rumus $O(\log n)$ dengan rincian $1 = 1, 2 = 2, 4 = 3, 10 = 4, 100 = 7, 1000 = 10$. Di mana log berasal dari pangkat log berbasis 2. Dengan begitu dapat mengetahui jumlah maksimal tebakan.

Untuk pola sendiri:

apabila ingin menebak angka 70

$a = \text{nilai tebakan pertama} // 2$

tebakan selanjutnya = nilai tebakan lebih dari a

jika hasil tebakan selanjutnya kurang dari, maka nilai yang dipakai tetap nilai lebih dari sebelumnya

$a = a // 2$

Simulasi

tebakan ke 1: 50 (mengambil nilai tengah) jawaban= "lebih dari itu"

tebakan ke 2: 75 (dari 50 + 25) jawaban = "kurang dari itu"

tebakan ke 3: 62 (dari 50 + 12) jawaban = "lebih dari itu"

tebakan ke 4: 68 (dari 62 + 6) jawaban = "lebih dari itu"

tebakan ke 5: 71 (dari 68 + 3) jawaban = "kurang dari itu"

tebakan ke 6: 69 (dari 68 + 1) jawaban = "lebih dari itu"

tebakan ke 7: antara 71 dan 69 hanya ada 1 angka = 70